PATENT ABSTRACTS OF JAPAN

(11)Publication number : **2000-057032**
(43)Date of publication of application : **25.02.2000**

(51)Int.Cl. **G06F 12/00**
G06F 17/30

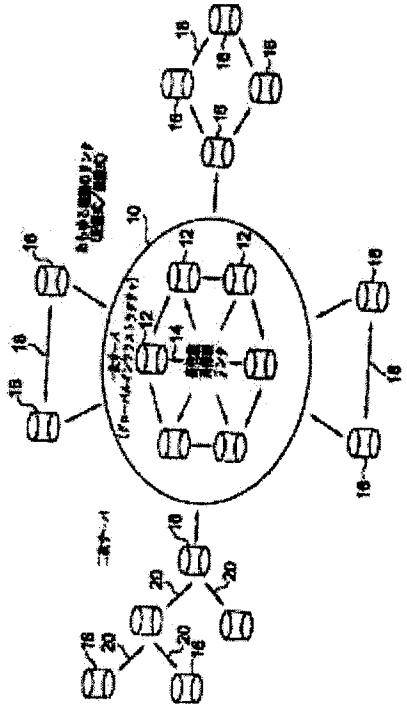
(21)Application number : **11-101694**
(22)Date of filing : **08.04.1999**

(71)Applicant : **MITSUBISHI ELECTRIC INF TECHNOL CENTER AMERICA INC**
(72)Inventor : **PENG LUOSCHENG**

(30)Priority
Priority number : **98 110748** Priority date : **03.07.1998** Priority country : **US**

(54) OBJECT SYNCHRONIZING GENERAL-PURPOSE SYSTEM USING PLURAL SERVERS, OBJECT SYNCHRONIZING SYSTEM AND METHOD USING TWO SERVERS, GENERAL-PURPOSE SYNCHRONIZING SYSTEM IN WHICH OBJECT EXISTING IN TWO SERVERS ARE SYNCHRONIZED REGARDLESS OF TYPE OF OBJECT AND METHOD FOR DETECTING AND ELIMINATING COMPETITION IN SYNCHRONIZATION OF OBJECTS IN TWO SERVERS

(57)Abstract:
PROBLEM TO BE SOLVED: To secure the synchronization of objects by means of plural servers by securing the synchronization between the object of a secondary server and that of a primary server to which the secondary server can be connected at the secondary server when a link is established between the said two objects.
SOLUTION: The primary servers 12 of a network 10 are connected to each other via a high performance link 14 which has higher reliability than a radio link in general. Many secondary servers 16 are linked (18) to each other in a pier-to-pier relation or linked (20) to each other in a hierarchical system. Then the data are automatically synchronized with each other among plural primary servers 12, and a 1st server 16 is connected to one of servers 12 via many servers 16 and a link. The object of the 1st server 16 is synchronized with the object of any one of servers 12 to which the server 16 can be connected at the server 16 when a link is established between these two servers.



(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号
特開2000－57032
（P2000－57032A）
(43)公開日　平成12年２月25日(2000.2.25)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｆ　12/00	５３３	Ｇ０６Ｆ　12/00	５３３Ｊ	
17/30		15/401	３４０Ｂ	

審査請求　未請求　請求項の数16　ＯＬ　（全 24 頁）

(21)出願番号　特願平11－101694

(22)出願日　平成11年４月８日(1999.4.8)

(31)優先権主張番号　０９／１１０７４８

(32)優先日　平成10年７月３日(1998.7.3)

(33)優先権主張国　米国（ＵＳ）

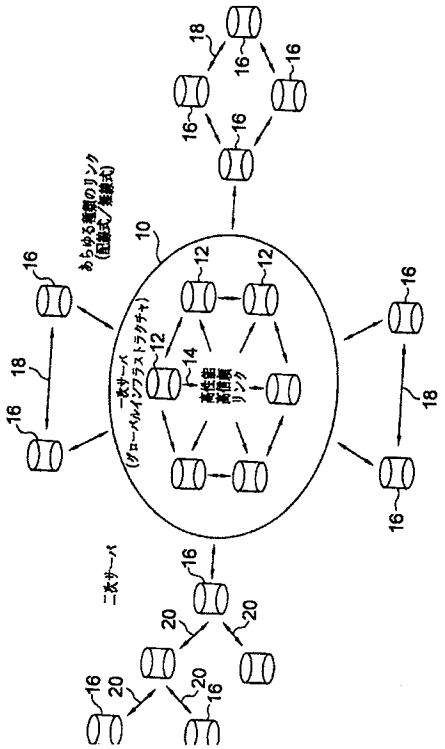
(71)出願人　597067574
ミツビシ・エレクトリック・インフォメイション・テクノロジー・センター・アメリカ・インコーポレイテッド
ＭＩＴＳＵＢＩＳＨＩ　ＥＬＥＣＴＲＩＣ　ＩＮＦＯＲＭＡＴＩＯＮ　ＴＥＣＨＮＯＬＯＧＹ　ＣＥＮＴＥＲ　ＡＭＥＲＩＣＡ，　ＩＮＣ．
アメリカ合衆国、マサチューセッツ州、ケンブリッジ、ブロードウエイ　201

(74)代理人　100057874
弁理士　曾我　道照　（外６名）

最終頁に続く

(54)【発明の名称】　複数のサーバでオブジェクトを同期させる汎用システム、２つのサーバでオブジェクトを同期させるシステム、２つのサーバでオブジェクトを同期させる方法、２つのサーバに存在するオブジ

(57)【要約】
【課題】　広域モバイルコンピューティングに適応し、サーバを同期させ、同時にプロセスをより効率的にするための汎用システムを提供する。
【解決手段】　二次サーバが概して信頼性の低いサーバによりリンクされた同期プロセスのバックボーンを形成する、一次サーバの高性能高信頼性リンクを用いたネットワークを備え、モバイル・コンピュータからの同期をクライアント／サーバ・モード及びピア・ツー・ピアの双方で行うことができ、二次サーバのいかなるトポロジをもサポートすることができるようにする。一次サーバは自動的にかつ頻繁に同期をとるが、二次サーバの同期は、意図しない同期を避けるユーザの制御下で行われる。また、要約バージョンベクトルを使用して、オブジェクト毎にバージョンベクトルを交換する必要を無くすことにより、転送されるデータの量を最小にする。



1

【特許請求の範囲】

【請求項１】　広域モバイルコンピューティングに適応する複数のサーバでオブジェクトを同期させる汎用システムであって、

複数の一次サーバのネットワークと、

上記複数の一次サーバを接続する高性能高信頼性リンクと、

上記複数の一次サーバにおいてデータを自動的に同期させる手段と、

多数の二次サーバと、

リンクによって第１の二次サーバを上記複数の一次サーバの１つに接続する手段と、

二次サーバのオブジェクトを、二次サーバが接続され得るいずれかの一次サーバのオブジェクトと、それらの間にリンクが確立した時に、二次サーバにおいて同期させる手段とを含むことを特徴とする複数のサーバでオブジェクトを同期させる汎用システム。

【請求項２】　第２の二次サーバと、上記第１及び第２の二次サーバ間の同期リンクとをさらに具備し、

クライアント／サーバ及びピア・ツー・ピア(peer to peer)の同期モードをサポートすることを特徴とする請求項１記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項３】　広域モバイルコンピューティングに適応する複数のサーバでオブジェクトを同期させる汎用システムであって、

複数の一次サーバのネットワークと、

上記一次サーバを接続する高性能高信頼性リンクと、

上記一次サーバにおいてオブジェクトを自動的に同期させる手段と、

多数の二次サーバと、

リンクによって、二次サーバを上記複数の一次サーバの１つに接続する手段と、

二次サーバにおいて二次サーバによって開始され、二次サーバのオブジェクトを、二次サーバが接続され得るいずれかの一次サーバのオブジェクトと同期させる手段とを具備し、

上記二次サーバ及び上記一次サーバの間の上記リンクの信頼性が低い場合でも、上記二次サーバが、接続され得る一次サーバとの間に高信頼性リンクを確立することが可能であると判断した時、同期が起こることを特徴とする複数のサーバでオブジェクトを同期させる汎用システム。

【請求項４】　上記同期させる手段は、上記二次サーバにおいて、上記リンクの信頼性を確認し、十分に信頼性の高いリンクを確立することができる場合にのみ、同期を開始する確認手段を含むことを特徴とする請求項３記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項５】　上記確認手段は、上記リンクにより情報

2

を転送するための帯域幅を決定する手段を含むことを特徴とする請求項４記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項６】　上記確認手段は、上記二次サーバで使用可能なリソースを確認する手段を含むことを特徴とする請求項４記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項７】　上記同期させる手段は、上記二次サーバで、上記同期を手動で開始する手段を有することを特徴とする請求項３記載の複数のサーバでオブジェクトを同期させる汎用システム。

【請求項８】　２つのサーバでオブジェクトを同期させるシステムであって、

２つの別々のロケーションにあるサーバと、

上記サーバをリンクするネットワークと、

上記サーバの１つにおいて、オブジェクト全体を基準とするか、又は差分を基準として、上記２つのサーバでオブジェクトを同期させる手段と、

上記同期させる手段に、オブジェクト全体の同期と差分同期とを切換えさせる手段とを含むことを特徴とする２つのサーバでオブジェクトを同期させるシステム。

【請求項９】　上記同期させる手段に切換えさせる手段は、上記サーバの１つにおいて使用可能なリソースを確認し、サーバのリソースが差分同期をサポートすることができない場合に、オブジェクト全体の同期に切換える手段を有することを特徴とする請求項８記載の２つのサーバでオブジェクトを同期させるシステム。

【請求項１０】　個々のオブジェクトのバージョンベクトルを交換する必要を無くすことにより、転送されるデータの量を最小にするように２つのサーバでオブジェクトを同期させる方法であって、

上記サーバの各々に、オブジェクトを包含するよう適合されたオブジェクトコンテナを供給するステップと、

サーバにおけるオブジェクトの各々の状態を要約する要約バージョンベクトルを各オブジェクトコンテナに供給するステップと、

第１のサーバから第２のサーバへ一つの要約バージョンベクトルのみを転送し、該第２のサーバにおけるオブジェクトに関連するバージョンベクトルが、該第１のサーバから受信した要約バージョンベクトルより新しいか又はそれと競合（conflict）する場合、あるいは、該第２のサーバにおける個々のオブジェクトの更新されたタイムスタンプが、該第１のサーバからの要約バージョンベクトルに関連したタイムスタンプより新しいか又はそれと競合する場合、即座にその第２のサーバから第１のサーバへ更新を返すことにより、同期を開始するステップとを含み、

上記要約バージョンベクトルは、上記オブジェクトコンテナの状態を要約すると共に、更新スタンプを有し、その更新スタンプの各々は、上記オブジェクトコンテナに

関連する識別子用のフィールドと、上記オブジェクトコンテナがオブジェクトを生成し、変更し又は削除した最後の時刻に対応するタイムスタンプ用のフィールドとを有し、上記タイムスタンプは、オブジェクトが生成され、変更され又は削除された時にそのオブジェクトコンテナによって生成されることを特徴とする２つのサーバでオブジェクトを同期させる方法。

【請求項１１】　上記同期を開始するステップは、上記要約バージョンベクトル及び更新スタンプに基づいて差分同期を実行して差分更新を生成し、それによって、差分更新によって示される、変更されているオブジェクトの部分のみを転送することによって、差分同期を行うステップを含むことを特徴とする請求項１０記載の２つのサーバでオブジェクトを同期させる方法。

【請求項１２】　最新共通バージョンベクトルを生成し、上記最新共通バージョンベクトルを使用して、選択された差分更新を一掃するステップをさらに含むことを特徴とする請求項１１記載の２つのサーバでオブジェクトを同期させる方法。

【請求項１３】　影響されていないサーバからのデータを用いて、前の失敗の地点から同期を再開するステップであって、前の失敗の地点と影響されていないサーバの存在を、そのサーバに関連する要約バージョンベクトルによって決定するステップをさらに含むことを特徴とする請求項１０記載の２つのサーバでオブジェクトを同期させる方法。

【請求項１４】　第１のサーバから第２のサーバへの更新に対応するオブジェクトのバージョンベクトルと、該第２のサーバの要約バージョンベクトルとの両方を、該第２のサーバが該第１のサーバから更新を受信した直後に更新し、それによって、その第１のサーバの要約バージョンベクトルを該第２のサーバの更新されたバージョンベクトルと比較することにより、該第２のサーバによってすでに受信されている更新を再送することなく、同期を復元することができるようにし、それによってきめの細かい同期を提供するステップをさらに含むことを特徴とする請求項１０記載の２つのサーバでオブジェクトを同期させる方法。

【請求項１５】　２つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期する汎用同期システムであって、

別々のロケーションで、オブジェクトのバージョンとそれに関連する更新ベクトルをそれぞれ有する複数のサーバと、

上記サーバをリンクするネットワークと、

オブジェクトのフォーマットに関係なく標準フォーマットでオブジェクトの更新を抽出する手段、及び、上記標準フォーマットのオブジェクトの更新を利用して標準プロトコルに基づいて上記複数のサーバでオブジェクトを同期させる手段を含む、サーバにおいて上記オブジェクトのセマンティクスを各上記同期から切離す手段とを具備し、

同期をオブジェクトの形態に関係なく行うことができ、それによって異なるシステム間での同期が可能であることを特徴とする２つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期する汎用同期システム。

【請求項１６】　第１のサーバから第２のサーバへ更新を伝搬し、２つのサーバでのオブジェクトの同期における競合を検出して解消する方法であって、

上記第１のサーバから上記第２のサーバへ要約バージョンベクトルを送信するステップと、

上記第１のサーバの要約バージョンベクトルを受信すると、上記第１のサーバに上記第２のサーバの要約バージョンベクトルを返信し、その後第２のサーバに存在し差分同期をサポートすることができるオブジェクトの識別子をすべて送信するステップと、

上記第２のサーバから要約バージョンベクトルと識別子とを受信すると、第１のサーバにおいて、受信した識別子から、上記第１のサーバに存在しないオブジェクトに対応する識別子を決定するステップと、

上記第１のサーバにおいて、上記２つのサーバの要約バージョンベクトルの共通するバージョンベクトルを計算するステップと、

上記第１のサーバにおいて、差分同期をサポートすることができるオブジェクトであって、そのバージョンベクトルについて、対応する差分更新がすべて無いベースが、上記計算された共通のバージョンベクトルより新しいオブジェクトの識別子をすべて決定し、オブジェクトの第１及び第２の決定された識別子が、同期において差分同期を実現することができず、そのためにオブジェクト全体の同期に切換えられなければならないオブジェクトの識別子であるステップと、

オブジェクト全体の同期に切換えなければならないオブジェクトの識別子を上記第２のサーバに送信するステップと、

上記第１のサーバから上記識別子を受信すると、上記第２のサーバにおいて、その２つのサーバの要約バージョンベクトルの共通するバージョンベクトルを計算するステップと、

上記第２のサーバにおいて、上記受信した識別子と異なるオブジェクトの識別子であり、差分同期をサポートすることができるオブジェクトであって、そのバージョンベクトルについて、対応する差分更新がすべて無いベースが、上記計算された共通のバージョンベクトルより新しいオブジェクトの識別子をすべて決定し、決定されたオブジェクトが、同期において差分同期を実現することができず、そのためにオブジェクト全体の同期に切換えられなければならないオブジェクトであるステップと、

上記第２のサーバから上記第１のサーバへ送信する場合

にオブジェクト全体で送信しなければならないオブジェクトの識別子のセットとして、上記決定された識別子に上記受信した識別子を加えるステップと、

上記第２のサーバにおいて、差分同期をサポートすることができないオブジェクト、及び、差分同期をサポートすることができ、その識別子が上記第２のサーバで決定された識別子のセットにあるオブジェクトを含む個々のオブジェクトのバージョンベクトルを、上記第１のサーバの要約バージョンベクトルと比較するステップと、

オブジェクト全体である更新として、上記第１のサーバの要約バージョンベクトルより新しいか又はそれと競合するバージョンベクトルを有するオブジェクトを抽出するステップと、

上記第２のサーバにおいて、上記第２のサーバによって決定された識別子のセットに無い識別子を有する該第２のサーバのオブジェクトに与えられた個々の差分更新の更新スタンプを比較するステップと、

上記第１のサーバの要約バージョンベクトルより新しいか又はそれと競合する更新スタンプを有する差分更新を抽出するステップと、

上記抽出した更新のすべてを、対応するオブジェクトの識別子及びバージョンベクトル又は更新スタンプと共に、上記第２のサーバから上記第１のサーバへ、一貫した順序で送信するステップと、

上記第１のサーバへ上記更新を送信し終わると、上記第２のサーバにおいて、第２のサーバの最新共通先祖バージョンベクトルと差分更新の一部又はすべてをパージするステップと、

上記第２のサーバから、対応するオブジェクトの識別子及びバージョンベクトル又は更新スタンプと共に更新を受信すると、受信した更新がオブジェクト全体であるか差分更新であるかを判断するステップと、

上記受信した更新がオブジェクト全体である場合、受信したバージョンベクトルを上記第１のサーバの対応するオブジェクトのバージョンベクトルと比較し、上記受信した更新が、該第１のサーバにおけるオブジェクトのバージョンベクトルより古いか又はそれと等しい場合、その受信したオブジェクトを破棄するステップと、

上記受信したバージョンベクトルが、上記第１のサーバのオブジェクトのバージョンベクトルより新しい場合、該第１のサーバのオブジェクトを上記受信したオブジェクトで置換えるか、あるいは、上記受信したバージョンベクトルが該第１のサーバのオブジェクトのバージョンベクトルと競合する場合、該第１のサーバのオブジェクトと上記受信したオブジェクトとの間に競合があることを識別し、該第１のサーバのオブジェクトにその競合を解消させるステップと、

上記受信した更新が差分更新である場合、上記受信した更新スタンプを上記第１のサーバの要約バージョンベクトルと比較するステップと

上記受信した更新スタンプが上記第１のサーバの要約バージョンベクトルより古いか又はそれと等しい場合、上記受信した差分更新を捨てるステップと、

あるいは、上記第２のサーバの要約バージョンベクトルを、上記第１のサーバの対応するオブジェクトに与えられた該第１のサーバの各差分更新の更新スタンプと比較するステップと、

上記第２のサーバの要約バージョンベクトルより新しいか又はそれと競合する更新スタンプを有する差分更新をすべて抽出するステップと、

上記第１のサーバで差分更新が抽出されない場合、該第１のサーバのオブジェクトに上記受信した差分更新を与え、あるいは、上記受信した差分更新が該第１のサーバで抽出された差分更新と競合することを識別するステップと、

上記第１のサーバのオブジェクトに上記競合を解消させるステップと、

上記第２のサーバから上記更新を受信し終わると、上記第１のサーバの最新共通先祖バージョンベクトルを更新するステップと、

上記第１のサーバにおいて上記差分更新の一部又はすべてをパージするステップとを含むことを特徴とする２つのサーバでのオブジェクトの同期における競合を検出して解消する方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】この発明は、広域モバイルコンピューティングに適応する複数のサーバでオブジェクトを同期させる汎用システムに係るもので、複数のサーバでオブジェクトを同期させる汎用システム、２つのサーバでオブジェクトを同期させるシステム、２つのサーバでオブジェクトを同期させる方法、２つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期される汎用同期システム、２つのサーバでのオブジェクトの同期における競合を検出して解消する方法に関する。

【０００２】

【従来の技術】ラップトップの形態のモバイルコンピューティングが急増し、あらゆる地点でデータのバージョンが生成されるに伴い、複数のロケーションで作成されるデータのバージョンを同期させる必要がある。このようなバージョンの同期は、個人がロケーションからロケーション、州から州、国から国へ移動するに従って、様々な異なったプラットフォームに亙って行われなければならない。

【０００３】複数のユーザが、文書又はデータの同じバージョンで作業をしている場合、従来は、文書を照合調整する様々な方法があった。あるユーザに対して別の文書が変更されたという事実を警告し、ユーザの判断で最新バージョンで更新するか、又は最新バージョンを拒否

するという、多くのシステムが提供されている。従来、バージョンが生成された時間を確認するためにログ・ファイルが使用されており、オブジェクトの複製（レプリカ）の間の競合を検出するために、長い間バージョンベクトルが利用されてきた。

【０００４】理解されるように、解消すべき更新の競合の検出方法にはいくつかある。すなわち、James J. Kistler等による"Disconnected Operation in the ＣＯＤＡ File System"（ACM Transactions on Computer Systems, 10(1), 1992）、Peter Reiher等による"Resolving File Conflicts in the Ficus File System"（USENIX Conference Proceedings, USENIX, June 1994）、Douglas B. Terry等による"Managing Update Conflicts in Bayou, a Weekly Connected Replicated Storage System"（Proceedings of the Fifth Symposium on Operating System Principles, ASM, December 1995）に述べられているような方法である。ＣＯＤＡ及びFicusシステムは、D. Stott Parker等による"Detection of Mutual Inconsistency in Distributed Systems"（IEEE Transactions on Software Engineering 9(3), May 1993）において最初に提案されているように、オブジェクト・レプリカ間の競合を検出するためにバージョンベクトルを利用する。一方、Bayouシステムは、それ自身で競合を検出せず、アプリケーションに頼っている。

【０００５】バージョンベクトルを利用して競合を検出する方法は、２つの基本的なパラダイムに発展してきている。１つのパラダイムでは、Ficusによって例示されているように、バージョンベクトルを利用して、個々のオブジェクト・レプリカの現在の状態について他のレプリカと比較して特徴を表す。ＣＯＤＡシステムによって推論することができるもう１つのパラダイムでは、バージョンベクトルを、オブジェクト・レプリカと、オブジェクトのセットを含むレプリケーション・ユニットとに適用している。

【０００６】

【発明が解決しようとする課題】ＣＯＤＡシステムは、ピア・ツー・ピア（peer to peer）同期をサポートしていないファイル・レプリケーション・システムであることが理解されよう。それは、本質的に、２つのクライアントが互いに直接同期することができないクライアント／サーバ・システムである。例えば、２人のモバイルユーザが互いのデータを交換したい場合、彼らが前もって互いを知らなければ、これはピア・ツー・ピア同期でなければならない。

【０００７】このような２つのクライアントがなんとかしてデータを同期させるためには、共通のサーバを介するという方法しかない。２つのクライアントが切断されると、リアルタイムに同期させることができなくなる。また、ＣＯＤＡシステムは、データを保存しＣＯＤＡサーバと同期する方法を採用しているため、モバイルコンピューティングには適していない。ＣＯＤＡでは、サーバがオープンした時、クライアントは常に、使用可能なすべてのサーバ中でのファイルの最新バージョンを得ようとする。ファイルがクローズした時、クライアントは常に、使用可能なすべてのサーバと同期しようとする。

【０００８】これは、大量のデータを転送するため、非常に費用がかかる。さらに、ＣＯＤＡでは、切断する前にファイル・アクセスがクライアントで保存されなかった場合、その切断されたオペレーションをサポートすることができない。これは、ＣＯＤＡでは、データが全体として複製されるのではなく、ファイルが逐一保存されるからである。このため、ＣＯＤＡシステムは、モバイルコンピューティングに適用しようとする場合、ファイルのオープン及びクローズの度に複数のサーバと並列同期をとる必要があるため、複雑である。

【０００９】Ficusシステムは、クライアント／サーバ同期をサポートしていないファイル・レプリケーション・システムである。それは、本質的にピア・ツー・ピア・システムであり、それ自体でレプリケーション・サイト間のすべての同期が自動化されている。これは、モバイルコンピューティング環境では経済的ではない。そのような環境では、通信リンクが高価で信頼性がないという特徴を有しているため、Ficusシステムは望ましくない。

【００１０】また、Ficusシステムでは、同期が自動的に行われるため、アプリケーションのセマンティクスが乱される可能性があり、それが原因で、このような場合にこのシステムが適用できなくなる。Ficusにおける同期は個々のファイルが保持するバージョンベクトルを使用することによって実現されるため、バージョンベクトルを各ファイルについて交換する必要がある。これは、無線通信には効率的でないと考えられる。例えば、ファイルのすべてのバージョンベクトルをサーバ間で転送しなければならないが、両方のサーバのファイルが同じである可能性がある。また、Ficusシステムは、同期プロセス中に全か無かの方式で全ファイルを送信するため、差分（differential）同期をサポートしていない。

【００１１】最後に、Bayouシステムは、データベースに適用される書込みのログを利用して、差分同期をサポートするデータ・ベース・レプリケーション・システムである。しかしながら、Bayouシステムでは、単一の一次サーバを使用して書込みの順序及びコミットメントを包括的に最終決定しなければならない。単一の一次サーバがダウンすると、書込みの最終的なコミットメントが遅れることになる。このため、一次サーバがダウンした場合、書込みの最終的な順序を確立できない。これによって、他のサーバの、安定したデータベースをできるだけ早く参照するという能力が遅くなる。

【００１２】さらに、Bayouシステムでは、アプリケーションが、書込みを行う度に競合の検出及び解消のコー

ドを加える必要がある。この要件により問題が生じる。第１に、競合の検出はシステムが行うのではなくアプリケーションが行うため、アプリケーションによってはそれが負担となる可能性がある。第２に、競合の検出及び解消のためのコードは、各書込みと共に伝搬しなければならず、それによって、データ同期のコスト及びトラフィックが著しく増大する可能性がある。さらに、上述したシステムはすべて、異なるシステムに亙って異なるタイプのデータを処理することができるほど柔軟ではない。

【００１３】この発明は上述した点に鑑みてなされたもので、上述した問題をすべて対処することができ、特にモバイルコンピューティング環境のためのシステムに適応する複数のサーバでオブジェクトを同期させる汎用システム、２つのサーバでオブジェクトを同期させるシステム、２つのサーバでオブジェクトを同期させる方法、２つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期される汎用同期システム、２つのサーバでのオブジェクトの同期における競合を検出して解消する方法を提供するものである。

【００１４】

【課題を解決するための手段】この発明に係る複数のサーバでオブジェクトを同期させる汎用システムは、複数の一次サーバのネットワークと、上記複数の一次サーバを接続する高性能高信頼性リンクと、上記複数の一次サーバにおいてデータを自動的に同期させる手段と、多数の二次サーバと、リンクによって第１の二次サーバを上記複数の一次サーバの１つに接続する手段と、二次サーバのオブジェクトを、二次サーバが接続され得るいずれかの一次サーバのオブジェクトと、それらの間にリンクが確立した時に、二次サーバにおいて同期させる手段とを含むことを特徴とするものである。

【００１５】また、第２の二次サーバと、上記第１及び第２の二次サーバ間の同期リンクとをさらに具備し、クライアント／サーバ及びピア・ツー・ピア(peer to peer)の同期モードをサポートすることを特徴とするものである。

【００１６】また、この発明に係る複数のサーバでオブジェクトを同期させる汎用システムは、複数の一次サーバのネットワークと、上記一次サーバを接続する高性能高信頼性リンクと、上記一次サーバにおいてオブジェクトを自動的に同期させる手段と、多数の二次サーバと、リンクによって、二次サーバを上記複数の一次サーバの１つに接続する手段と、二次サーバにおいて二次サーバによって開始され、二次サーバのオブジェクトを、二次サーバが接続され得るいずれかの一次サーバのオブジェクトと同期させる手段とを具備し、上記二次サーバ及び上記一次サーバの間の上記リンクの信頼性が低い場合でも、上記二次サーバが、接続され得る一次サーバとの間に高信頼性リンクを確立することが可能であると判断した時、同期が起こることを特徴とするものである。

【００１７】また、上記同期させる手段は、上記二次サーバにおいて、上記リンクの信頼性を確認し、十分に信頼性の高いリンクを確立することができる場合にのみ、同期を開始する確認手段を含むことを特徴とするものである。

【００１８】また、上記確認手段は、上記リンクにより情報を転送するための帯域幅を決定する手段を含むことを特徴とするものである。

【００１９】また、上記確認手段は、上記二次サーバで使用可能なリソースを確認する手段を含むことを特徴とするものである。

【００２０】また、上記同期させる手段は、上記二次サーバで、上記同期を手動で開始する手段を有することを特徴とするものである。

【００２１】また、この発明に係る２つのサーバでオブジェクトを同期させるシステムは、、２つの別々のロケーションにあるサーバと、上記サーバをリンクするネットワークと、上記サーバの１つにおいて、オブジェクト全体を基準とするか、又は差分を基準として、上記２つのサーバでオブジェクトを同期させる手段と、上記同期させる手段に、オブジェクト全体の同期と差分同期とを切換えさせる手段とを含むことを特徴とするものである。

【００２２】また、上記同期させる手段に切換えさせる手段は、上記サーバの１つにおいて使用可能なリソースを確認し、サーバのリソースが差分同期をサポートすることができない場合に、オブジェクト全体の同期に切換える手段を有することを特徴とするものである。

【００２３】また、この発明に係る２つのサーバでオブジェクトを同期させる方法は、個々のオブジェクトのバージョンベクトルを交換する必要を無くすことにより、転送されるデータの量を最小にするように２つのサーバでオブジェクトを同期させる方法であって、上記サーバの各々に、オブジェクトを包含するよう適合されたオブジェクトコンテナを供給するステップと、サーバにおけるオブジェクトの各々の状態を要約する要約バージョンベクトルを各オブジェクトコンテナに供給するステップと、第１のサーバから第２のサーバへ一つの要約バージョンベクトルのみを転送し、該第２のサーバにおけるオブジェクトに関連するバージョンベクトルが、該第１のサーバから受信した要約バージョンベクトルより新しいか又はそれと競合（conflict）する場合、あるいは、該第２のサーバにおける個々のオブジェクトの更新されたタイムスタンプが、該第１のサーバからの要約バージョンベクトルに関連したタイムスタンプより新しいか又はそれと競合する場合、即座にその第２のサーバから第１のサーバへ更新を返すことにより、同期を開始するステップとを含み、上記要約バージョンベクトルは、上記オブジェクトコンテナの状態を要約すると共に、更新スタ

ンプを有し、その更新スタンプの各々は、上記オブジェクトコンテナに関連する識別子用のフィールドと、上記オブジェクトコンテナがオブジェクトを生成し、変更し又は削除した最後の時刻に対応するタイムスタンプ用のフィールドとを有し、上記タイムスタンプは、オブジェクトが生成され、変更され又は削除された時にそのオブジェクトコンテナによって生成されることを特徴とするものである。

【００２４】また、上記同期を開始するステップは、上記要約バージョンベクトル及び更新スタンプに基づいて差分同期を実行して差分更新を生成し、それによって、差分更新によって示される、変更されているオブジェクトの部分のみを転送することによって、差分同期を行うステップを含むことを特徴とするものである。

【００２５】また、最新共通バージョンベクトルを生成し、上記最新共通バージョンベクトルを使用して、選択された差分更新を一掃するステップをさらに含むことを特徴とするものである。

【００２６】また、影響されていないサーバからのデータを用いて、前の失敗の地点から同期を再開するステップであって、前の失敗の地点と影響されていないサーバの存在を、そのサーバに関連する要約バージョンベクトルによって決定するステップをさらに含むことを特徴とするものである。

【００２７】また、第１のサーバから第２のサーバへの更新に対応するオブジェクトのバージョンベクトルと、該第２のサーバの要約バージョンベクトルとの両方を、該第２のサーバが該第１のサーバから更新を受信した直後に更新し、それによって、その第１のサーバの要約バージョンベクトルを該第２のサーバの更新されたバージョンベクトルと比較することにより、該第２のサーバによってすでに受信されている更新を再送することなく、同期を復元することができるようにし、それによってきめの細かい同期を提供するステップをさらに含むことを特徴とするものである。

【００２８】また、この発明に係る２つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期する汎用同期システムは、別々のロケーションで、オブジェクトのバージョンとそれに関連する更新ベクトルをそれぞれ有する複数のサーバと、上記サーバをリンクするネットワークと、オブジェクトのフォーマットに関係なく標準フォーマットでオブジェクトの更新を抽出する手段、及び、上記標準フォーマットのオブジェクトの更新を利用して標準プロトコルに基づいて上記複数のサーバでオブジェクトを同期させる手段を含む、サーバにおいて上記オブジェクトのセマンティクスを各上記同期から切離す手段とを具備し、同期をオブジェクトの形態に関係なく行うことができ、それによって異なるシステム間での同期が可能であることを特徴とするものである。

【００２９】さらに、この発明に係る２つのサーバでのオブジェクトの同期における競合を検出して解消する方法は、第１のサーバから第２のサーバへ更新を伝搬し、２つのサーバでのオブジェクトの同期における競合を検出して解消する方法であって、上記第１のサーバから上記第２のサーバへ要約バージョンベクトルを送信するステップと、上記第１のサーバの要約バージョンベクトルを受信すると、上記第１のサーバに上記第２のサーバの要約バージョンベクトルを返信し、その後第２のサーバに存在し差分同期をサポートすることができるオブジェクトの識別子をすべて送信するステップと、上記第２のサーバから要約バージョンベクトルと識別子とを受信すると、第１のサーバにおいて、受信した識別子から、上記第１のサーバに存在しないオブジェクトに対応する識別子を決定するステップと、上記第１のサーバにおいて、上記２つのサーバの要約バージョンベクトルの共通するバージョンベクトルを計算するステップと、上記第１のサーバにおいて、差分同期をサポートすることができるオブジェクトであって、そのバージョンベクトルについて、対応する差分更新がすべて無いベースが、上記計算された共通のバージョンベクトルより新しいオブジェクトの識別子をすべて決定し、オブジェクトの第１及び第２の決定された識別子が、同期において差分同期を実現することができず、そのためにオブジェクト全体の同期に切換えられなければならないオブジェクトの識別子であるステップと、オブジェクト全体の同期に切換えなければならないオブジェクトの識別子を上記第２のサーバに送信するステップと、上記第１のサーバから上記識別子を受信すると、上記第２のサーバにおいて、その２つのサーバの要約バージョンベクトルの共通するバージョンベクトルを計算するステップと、上記第２のサーバにおいて、上記受信した識別子と異なるオブジェクトの識別子であり、差分同期をサポートすることができるオブジェクトであって、そのバージョンベクトルについて、対応する差分更新がすべて無いベースが、上記計算された共通のバージョンベクトルより新しいオブジェクトの識別子をすべて決定し、決定されたオブジェクトが、同期において差分同期を実現することができず、そのためにオブジェクト全体の同期に切換えられなければならないオブジェクトであるステップと、上記第２のサーバから上記第１のサーバへ送信する場合にオブジェクト全体で送信しなければならないオブジェクトの識別子のセットとして、上記決定された識別子に上記受信した識別子を加えるステップと、上記第２のサーバにおいて、差分同期をサポートすることができないオブジェクト、及び、差分同期をサポートすることができ、その識別子が上記第２のサーバで決定された識別子のセットにあるオブジェクトを含む個々のオブジェクトのバージョンベクトルを、上記第１のサーバの要約バージョンベクトルと比較するステップと、オブジェクト全体である更

新として、上記第１のサーバの要約バージョンベクトルより新しいか又はそれと競合するバージョンベクトルを有するオブジェクトを抽出するステップと、上記第２のサーバにおいて、上記第２のサーバによって決定された識別子のセットに無い識別子を有する該第２のサーバのオブジェクトに与えられた個々の差分更新の更新スタンプを比較するステップと、上記第１のサーバの要約バージョンベクトルより新しいか又はそれと競合する更新スタンプを有する差分更新を抽出するステップと、上記抽出した更新のすべてを、対応するオブジェクトの識別子及びバージョンベクトル又は更新スタンプと共に、上記第２のサーバから上記第１のサーバへ、一貫した順序で送信するステップと、上記第１のサーバへ上記更新を送信し終わると、上記第２のサーバにおいて、第２のサーバの最新共通先祖バージョンベクトルと差分更新の一部又はすべてをパージするステップと、上記第２のサーバから、対応するオブジェクトの識別子及びバージョンベクトル又は更新スタンプと共に更新を受信すると、受信した更新がオブジェクト全体であるか差分更新であるかを判断するステップと、上記受信した更新がオブジェクト全体である場合、受信したバージョンベクトルを上記第１のサーバの対応するオブジェクトのバージョンベクトルと比較し、上記受信した更新が、該第１のサーバにおけるオブジェクトのバージョンベクトルより古いか又はそれと等しい場合、その受信したオブジェクトを破棄するステップと、上記受信したバージョンベクトルが、上記第１のサーバのオブジェクトのバージョンベクトルより新しい場合、該第１のサーバのオブジェクトを上記受信したオブジェクトで置換えるか、あるいは、上記受信したバージョンベクトルが該第１のサーバのオブジェクトのバージョンベクトルと競合する場合、該第１のサーバのオブジェクトと上記受信したオブジェクトとの間に競合があることを識別し、該第１のサーバのオブジェクトにその競合を解消させるステップと、上記受信した更新が差分更新である場合、上記受信した更新スタンプを上記第１のサーバの要約バージョンベクトルと比較するステップと、上記受信した更新スタンプが上記第１のサーバの要約バージョンベクトルより古いか又はそれと等しい場合、上記受信した差分更新を捨てるステップと、あるいは、上記第２のサーバの要約バージョンベクトルを、上記第１のサーバの対応するオブジェクトに与えられた該第１のサーバの各差分更新の更新スタンプと比較するステップと、上記第２のサーバの要約バージョンベクトルより新しいか又はそれと競合する更新スタンプを有する差分更新をすべて抽出するステップと、上記第１のサーバで差分更新が抽出されない場合、該第１のサーバのオブジェクトに上記受信した差分更新を与え、あるいは、上記受信した差分更新が該第１のサーバで抽出された差分更新と競合することを識別するステップと、上記第１のサーバのオブジェクトに上記競合を解消

させるステップと、上記第２のサーバから上記更新を受信し終わると、上記第１のサーバの最新共通先祖バージョンベクトルを更新するステップと、上記第１のサーバにおいて上記差分更新の一部又はすべてをパージするステップとを含むことを特徴とするものである。

【００３０】

【発明の実施の形態】以下、この発明を詳細に説明する。この発明によれば、多くの場合中央サーバ又はピア・ツー・ピア・システムいずれかへのリンクが信頼できないモバイルコンピューティング環境において、異なるサーバに格納されたオブジェクトのバージョンに対し信頼性のある同期をとるシステムが提供される。

【００３１】このシステムは、単一の中央サーバ又はピア・ツー・ピア・サーバ・システムの代りに、二次サーバの同期をとるための高性能高信頼性リンクにリンクされた一次サーバのネットワークを用いる。複数の一次サーバを用いるため、データを、より優れたデータ整合性及び信頼性をもって、より広いエリアに分配することができる。モバイル・コンピュータを二次サーバとすることにより、これらの信頼性の低いサーバを、同期プロセスのバックボーンであるサーバから区別する。

【００３２】一実施形態では、一次サーバは、一次サーバ間で優れたデータ整合性を維持するために、自動的にかつ頻繁に同期がとられる。一方、二次サーバからの信頼性の低い同期は、二次サーバから制御され、それによって通信コストを低減することができ、文書又は他のデータは、二次サーバでユーザが許可しなければ同期がとられない。これによって、ユーザは、文書のドラフトや、終了するまで同期がとられるべきではない未完成のソフトウエア又はプログラミングを用いて作業することができる。

【００３３】重要な特徴として、本システムでは、要約（summarizing）バージョンベクトルを用いて、同期がとられている２つのオブジェクトに差があるか否かに関わらず、個々のオブジェクトについてバージョンベクトルを交換する必要を無くすことにより、同期プロセス中に転送されるデータの量を最小限にする。

【００３４】ここで用いる「要約バージョンベクトル」という語は、サーバでオブジェクトコンテナの状態を要約するフィールドを有するベクトルを意味する。各要約バージョンベクトルは、更新スタンプのベクトルである。各更新スタンプは、関連するオブジェクトコンテナの識別子についてのフィールドと、関連するタイムスタンプについてのフィールドとを有している。

【００３５】オブジェクトが同じでない場合、従来技術では、第１のロケーションにある各オブジェクトについてのバージョンベクトルが第２のロケーションに送信される時、対応する更新が第１のロケーションに返信されていた。このため、不要なデータが転送されるという状態が発生していた。

15

【００３６】本システムでは、単一の要約バージョンベクトルが第１のロケーションから第２のロケーションに送信され、その後全ての更新が即座に第１のロケーションに返される。一実施形態では、要約バージョンベクトルが第２のロケーションで調べられ、第１のロケーションに、オブジェクト全体を送信する必要があるか、又はオブジェクトの一部を送信する必要があるかが判断される。これは、バージョンベクトルをオブジェクト全体に対して定義するか、又はバージョンベクトルを、オブジェクトのベース及びその差分更新各々についての更新スタンプに対して定義するかによって行われる。

【００３７】ここで、オブジェクトのベースとは、差分更新の影響を受けた変化がまったく無い初期形態にあるオブジェクトを言う。

【００３８】次に、受信した要約バージョンベクトルは、個々のオブジェクトの上記バージョンベクトル、又は個々の差分更新についての更新スタンプのいずれかと比較される。上述したように、差分更新の更新スタンプは、それを生成したオブジェクトコンテナの識別子と、それが生成された時のタイムスタンプからなる。

【００３９】オブジェクト全体又は差分更新はすべて、それらのバージョンベクトル又は更新されたタイムスタンプが、受信した要約バージョンベクトルより新しいか、もしくはそれと競合する場合においてのみ、第１のロケーションに送信される。これにより、従来技術のシステムに関連する不要なバージョンベクトルの転送を無くすことができる。このため、要約バージョンベクトル・システムにより、個々のオブジェクトに関するバージョンベクトルを繰り返して転送することがなくなる。

【００４０】要約バージョンベクトルの他の用途は、以下の通りである。なお、従来、同期システムは全か無かの基準で動作しており、それは、同期が完了前に失敗した場合に、すべてのデータが存在しなければならないことを意味していた。本システムでは、更新が第１のサーバから第２のサーバに送信される時、更新の受信が成功した直後に、対応するオブジェクトのバージョンベクトルと第２のサーバの要約バージョンベクトルの両方が更新される。

【００４１】従って、同期が失敗した場合、第１のサーバの要約バージョンベクトルを第２のサーバの更新されたバージョンベクトルと比較することにより、前回の同期で第２のサーバから既に受信している更新を再送することなく、同期が復元される。このようなシステムでは、よりきめ細かい同期が提供され、また、そのため、このようなシステムはより耐障害性が優れている。

【００４２】これは、本システムでは、転送されるデータのユニットが、差分更新（サイズが小さいためアトムと呼ばれる）であるという理由による。これは、転送データのユニット毎にオブジェクト全体を転送しなければならない従来技術のシステムとは異なっている。３つ以

16

上のサーバを有する場合、第3のサーバが、障害の時刻から再開の時刻までの間、第２のサーバと同期しているのならば、同期は、障害地点の後で第１のサーバ及び第２のサーバ間で復元される。これは、要約バージョンベクトルを用いることによって可能になる。

【００４３】コンテナ内のオブジェクトは、例えば、文書、プログラム、リレーショナル・データベースのテーブルの行等、どのようなオブジェクトでもよく、そのため、本システムは汎用システムとなっていることが理解されよう。本システムは、様々なデータの形式に対しての同期プロセスを組込んでおり、それは、オブジェクトのセマンティクスを同期化から切離すことによって可能となっている。これは、データの形式に依存している既存のシステムとは対照的である。

【００４４】この発明の重要な態様として、本システムでは、データのセマンティクスを同期化から切離しているため、Word文書、McDraw文書及びFrameMaker文書の間の同期化のように異なるシステム間の同期が可能である。これは、標準フォーマットで更新を抽出し、標準プロトコルに基づいて同期をとることによって行われる。

【００４５】動作中に、あるフォーマットのあるロケーションでの更新が、標準プロトコルでの更新に変換される。標準プロトコルでの更新は、第２のロケーションに転送され、そこで第２のロケーションのオブジェクトのフォーマットに変換される。

【００４６】さらに、本システムでは、データ転送を最小化するため、オブジェクトのバージョン間で変更されるデータのみに対して差分同期を使用する。この発明の固有の特徴として、本システムは、例えばローカル・メモリがいっぱいになった時、あるいはディスク・スペースが制限されている時に、オブジェクト全体の同期から差分同期へ切換える。さらに、あるロケーションのオブジェクトが非常に古い場合、差分同期が用いられているならば、オブジェクト全体または全部の同期に自動的にシフトすることによって、第２のロケーションのオブジェクト全体を更新するのが便利である。

【００４７】この発明のさらなる特徴としては、モバイル・コンピュータからの同期を、クライアント／サーバ・モードにおいて二次サーバから一次サーバへ行うこともでき、また、ピア・ツー・ピア・システム又は階層システムのいずれかにおいて二次サーバ間で行うこともでき、これによってモバイル間同期が可能となることである。例えば、ある二次サーバがユーザのデスクトップ・コンピュータであり、他の二次サーバが二次的用途のためのモバイル装置である場合、この２つの二次サーバ間のデータ同期はクライアント／サーバ型であり、２つのモバイル装置のデータ同期（例えば、２人のセールスマンが所有し互いのセールス・データを交換する場合）は、ピア・ツー・ピア型で行う。本システムは、ピア・ツー・ピア構造及び階層構造の双方をサポートしている

結果、二次サーバのいかなるトポロジをもサポートする。

【００４８】サーバは、しばしば、選択された数のサーバからのデータのみに関わるため、システム内のすべてのサーバと同期をとる必要はない。多くのサーバを有し、同期させたいデータを有するのはその一部のみであるようなシステムでは、すべてのオブジェクト及びすべての更新を追跡しようとすると、メモリが急速に使い果たされる。

【００４９】この発明の一実施形態でこの問題を解決するために、選択されたサーバのグループにおいて、更新及びバージョンの変更を選択的にパージするために、最新共通先祖バージョンベクトルを利用する。そのような更新及びバージョンの変更は、この最新共通先祖バージョンベクトルより古いか又はそれと等しいものである。また、上記パージは、問題のサーバ、例えば選択されたサーバのグループに伝搬された差分更新又は削除されたオブジェクトをパージすることによって行う。

【００５０】理想的には、システム内のすべてのサーバが更新を受信した場合においてのみ、サーバが差分更新又は削除されたオブジェクトをパージするべきである。しかしながら、サーバ数、特に二次サーバ数は動的に変化し、非常に長い間システム全体から切断されるサーバもあるため、すべてのサーバが更新を受信した後にのみ更新をパージすることは実用的ではなく、メモリ又はディスク・スペースが、実際に更新をパージすることが可能となる前に、使い果たされてしまうことがある。

【００５１】従って、本システムでは、サーバが、サーバのセットを選択して、選択されたサーバのすべてに更新が伝搬され終わった場合に、更新をパージすることができる。さらに、選択されたサーバは、予め設定された制限時間を超えて、選択しているサーバと同期していない場合、更新をパージする際に考慮の対象外とすることができる。

【００５２】本システムでは、この更新のパージを、最新共通先祖バージョンベクトルを利用して行う。サーバは、選択されたサーバのいずれかと同期する場合、選択されたサーバのバージョンベクトルをローカルに記録する。そして、最新共通先祖バージョンベクトルは、それ自身のバージョンベクトルと選択されたすべてのサーバのバージョンベクトルとから計算される。

【００５３】それらの更新スタンプ又はバージョンベクトルが、計算された最新共通バージョンベクトルより新しいか又はそれと等しい場合、すべての差分更新又は削除されたオブジェクトの情報がパージされる。以下に示す同期アルゴリズムは、要約バージョンベクトル及び更新スタンプの双方を用いて、転送されるデータを最小化する。同期中には、本アルゴリズムは、サーバからサーバへの更新の伝搬を実現するために、以下にリストしたステップを実行する。

【００５４】第１のサーバは、その要約バージョンベクトルを第２のサーバに送信する。第２のサーバは、第１のサーバの要約バージョンベクトルを受信すると、自身の要約バージョンベクトルを第１のサーバに送信し、その後第２のサーバ内に存在し、差分同期をサポートすることができるすべてのオブジェクトの識別子を送信する。

【００５５】第１のサーバは、第２のサーバから要約バージョンベクトルと識別子を受信すると、オブジェクト全体として受信し第２のサーバに送信する必要のあるオブジェクトの識別子をすべて見つけ出す。

【００５６】このステップのサブ・ステップは以下の通りである。

ａ）第２のサーバ内に存在するが第１のサーバ内には存在しないすべて　のオブジェクトを探し出し、それらの識別子を、第２のサーバから受信する必要のあるオブジェクトのリストに加える。これは、受信したオブジェクトの識別子で第１のサーバ内に存在するオブジェクトの識別子と一致するものがあるか否かをチェックすることによって行う。

【００５７】ｂ）両方のサーバに存在し、差分同期をサポートすることができるすべてのオブジェクトを探し出す。そして、これらのオブジェクトのベース・バージョンベクトルで２つのサーバの共通バージョンベクトルより新しいものがあるか否かをチェックする。ここで、オブジェクトのベース・バージョンベクトルとは、差分更新がまったくないオブジェクトのバージョンベクトルのことを言い、共通バージョンベクトルとは、２つのサーバの要約バージョンベクトルがそこから分岐した状態を反映するバージョンベクトルのことを言う。また、チェックをパスしたオブジェクトの識別子はすべて、第２のサーバから受信する必要のあるオブジェクトのリストに加えられる。

【００５８】ｃ）サブ・ステップａ）及びｂ）から得られる識別子をすべて第２のサーバに送信する。

【００５９】第２のサーバは、第１のサーバから識別子を受信すると、その最新共通先祖バージョンベクトルを第１のサーバの要約バージョンベクトルと比較する。第１のサーバの要約バージョンベクトルが第２のサーバの最新共通先祖バージョンベクトルより古い場合、それが含む個々のオブジェクト全体をすべて抽出する。そうでなければ、その要約バージョンベクトルと第１のサーバの要約バージョンベクトルとの比較を続ける。

【００６０】第２のサーバの要約バージョンベクトルが第１のサーバの要約バージョンベクトルより新しい場合、第２のサーバは、その中に存在しているすべてのオブジェクトをスキャンする。オブジェクトの識別子が第１のサーバから受信した識別子のリストにある場合、それは全体として第１のサーバに送信される。オブジェクトが差分同期をサポートせず　そのバージョンベクトル

が第１のサーバの要約バージョンベクトルより新しいか又はそれと競合するため、そのオブジェクトが全体として送信する必要のあるオブジェクトである場合、このオブジェクトもまた、第１のサーバに送信される。

【００６１】オブジェクトが差分同期をサポートするオブジェクトであり、そのベース・バージョンベクトルが第１のサーバの要約バージョンベクトルより新しい場合、このオブジェクトもまた、全体として第１のサーバに送信される。最後に、オブジェクトが差分同期をサポートするオブジェクトであり、その更新のスタンプが第１のサーバの要約バージョンベクトルより新しいか又はそれと競合する場合、これらの更新は、差分更新として第１のサーバに送信される。送信されているオブジェクト又は更新のすべての順序は、バージョンベクトルの順序付け又は関連する更新スタンプによってまず決定される。順序付けに矛盾がある場合、二次的な順序付けプロセスが開始する。

【００６２】ステップ４が終了した後、第２のサーバは、第１のサーバが第２のサーバが選択したサーバのうちの１つである場合、以前に格納していた第１のサーバの要約バージョンベクトルを格納又は更新する。また、選択されたサーバの要約バージョンベクトルがすべて第２のサーバに格納された場合、その最新共通先祖バージョンベクトルを再計算する。また、それによって、バージョンベクトル又は更新スタンプが最新共通先祖バージョンベクトルより古いか又はそれと等しいすべての削除されたオブジェクトの情報又は差分更新を一掃ジする。

【００６３】第１のサーバは、第２のサーバから各オブジェクト及び更新を受信すると、以下のように情報を更新する。

ａ）受信したオブジェクト又は更新が、第１のサーバ内の対応するオブジェクトのバージョンベクトルより古いか又はそれと等しいバージョンベクトル又はタイムスタンプを有する場合、このオブジェクト又は更新は、破棄される。

【００６４】ｂ）オブジェクト全体が受信され、そのバージョンベクトルが第１のサーバ内の対応するオブジェクトのバージョンベクトルより新しい場合、第１のサーバ内のオブジェクトは、受信されたオブジェクトに置換えられ、それによって、オブジェクトのバージョンベクトル及び第１のサーバの要約バージョンベクトルもまた更新される。

【００６５】ｃ）オブジェクト全体が受信され、そのバージョンベクトルが第１のサーバ内の対応するオブジェクトのバージョンベクトルと競合している場合、オブジェクトの２つのバージョンは、第１のサーバ内のオブジェクトについての照合調整メソッドを呼出すことによって、アーギュメントとして受信されたオブジェクトと調合調整される。その結果、第１のサーバ内のオブジェクトを置換えるために新たなオブジェクトが生成され、それによって、新たなオブジェクトについての新たなバージョンベクトルが生成されて、第１のサーバの要約バージョンベクトルが更新される。

【００６６】ｄ）差分更新が受信され、その更新スタンプ及び受信した第２のサーバの要約バージョンベクトルが、その差分更新がまだ第１のサーバ内に無く、第１のサーバにおいて対応するオブジェクトの差分更新後に生成されたことを示す場合、受信された更新は、第１のサーバ内のオブジェクトの差分更新のトップに加えられる。このように、第１のサーバの要約バージョンベクトルは、更新される。

【００６７】ｅ）差分更新が受信され、その更新スタンプ及び受信した第２のサーバの要約バージョンベクトルが、差分更新がまだ第１のサーバ内に無く、第１のサーバにおいて対応するオブジェクトの差分更新後に生成されなかったことを示す場合、２つのサーバの共通する状態の後で生成された第１のサーバ内の差分更新が抽出され、抽出された差分更新及び受信した差分更新の両方が、それらの間の競合を照合調整するために、第１のサーバ内のオブジェクトに関する照合調整メソッドに対し、アーギュメントとして渡される。その結果、第１のサーバ内のオブジェクトは、新たなオブジェクトに置換えられるか、又は新たな更新が第１のサーバ内のオブジェクトの差分更新に加えられる。

【００６８】このようにして、新たなオブジェクトの新たなバージョンベクトルか又は新たな更新についての更新スタンプが生成され、第１のサーバの要約バージョンベクトルが更新される。

【００６９】ステップ６の終了後、第２のサーバが第１のサーバが選択したサーバのうちの１つである場合、第１のサーバは、以前に格納していた第２のサーバの要約バージョンベクトルを格納又は更新する。また、選択されたサーバの要約バージョンベクトルがすべて第１のサーバ内に格納されている場合、その最新共通先祖バージョンベクトルを再計算し、そのようにして、バージョンベクトル又は更新スタンプが最新共通先祖バージョンベクトルより古いか又はそれと等しい削除されたオブジェクトの情報又は差分更新をすべてパージする。

【００７０】ステップ７の終了後、第２のサーバから第１のサーバへの更新伝搬が完了する。両方向の更新伝搬が必要である場合、第１のサーバを第２のサーバで置換え、またその逆を行うことにより、ステップ１）からステップ７）を繰り返すことができる。

【００７１】上記説明から分かるように、本アルゴリズムには、既存のアルゴリズムと比較して２つの利点がある。第１に、サーバからサーバへ、オブジェクト毎のバージョンベクトルを転送しなくてよい。第２に、あるサーバには取込まれているが他のサーバには取込まれていない差分更新を転送することができる。本アルゴリズムの更なる利点は、より きめの細かい同期を提供し、か

つ、より耐障害性に優れているということである。

【００７２】要約すると、広域モバイルコンピューティングに適応し、同時に同期プロセスをより効率的にするサーバを同期させるための汎用システムを提供する。第１に、高性能高信頼性リンクを有する複数の一次サーバを導入することにより、優れたデータ可用性及び拡張性が確実になる。第２に、二次サーバを導入することにより、本システムが、信頼性の低い高価なリンクを使用するモバイルコンピューティングに適するようになる。また、二次サーバによって同期を制御することにより、同期を経済的に行うことができる。さらに、自動的な同期を行うことができないアプリケーションがあるため、意図しない同期を防ぐ同期のユーザ制御がクリティカルである。

【００７３】この発明の第２の態様は、要約バージョンベクトルを使用することである。この要約バージョンベクトルを使用して、個々のオブジェクトのバージョンベクトルを交換する必要をなくすことによって、同期プロセスにおいて転送されるデータの量を最小限にする。

【００７４】また、個々の差分更新に関連した更新スタンプを用いて動作して差分同期を可能にすることにより、転送されるデータをさらに減らすことができる。サーバの要約バージョンベクトル及びその選択されたサーバのセットの要約バージョンベクトルから、最新共通バージョンベクトルを構成することができ、それを用いてサーバ上の差分更新を減らすことができる。

【００７５】さらに、要約バージョンベクトルによって、影響を受けていないサーバからのデータを用いて、以前の失敗の地点から同期を再開することができる。また、転送されるデータのアトムとしての差分更新を可能にすることにより、きめの細かい同期をサポートする。

【００７６】この発明の重要な特徴として、システムは、オブジェクト全体の同期と差分同期とを自動的に切換える。例えば、メモリ管理において、差分処理がメモリに対して集中する性質を有するために、差分同期からオブジェクト全体の同期へ切換えることがしばしば望ましい。また、オブジェクト全体の同期を行っている場合、転送されるデータを最少にするために差分同期に切換えることが有益な場合もある。

【００７７】最後に、これも重要なことであるが、本システムでは、データのセマンティクスが同期化から切離しているため、本システムにより、異なるシステム間の同期が可能となり、これによりシステムがデータのフォーマットに関して汎用的なものとなる。

【００７８】この発明のこれらの特徴及び他の特徴は、図面と共に詳細な説明を参照してより理解されるであろう。

【００７９】図１では、一次及び二次サーバを有するシステム・アーキテクチャを示す。ここでは、一次サーバ１２のネットワーク１０が、一般に無線リンクより桁違いに信頼性の高い高性能リンク１４を利用して、相互接続されている。高性能リンクを使用する目的は、同期保全性を確実にすることと、二次サーバの一次サーバへの接続を可能にすることである。

【００８０】図示しているように、多数の二次サーバ１６が、１８に示すようにピア・ツー・ピア関係で互いにリンクされているか、又は、２０に示すように階層方式でリンクされている。上述したように、この一次サーバ／二次サーバ・アーキテクチャは、実質的にあらゆる型のシステム（ピア・ツー・ピア・システムでも階層システムでも）との高信頼の同期を行うことができる。一次サーバ／二次サーバ・アーキテクチャを有するために、あらゆる型のシステムを適応ことができ、それによって、本システムは二次サーバ間のあらゆるトポロジに適応することができる。

【００８１】図２において、一実施形態では、本システムのサーバのアーキテクチャは、多数のオブジェクトコンテナ２２と、それに接続されたオブジェクトコンテナ・マネージャ２４を含んでいる。オブジェクトコンテナ・マネージャ２４は、シンクロナイザ・マネージャ２６に接続されており、シンクロナイザ・マネージャ２６は、オブジェクトコンテナ２２及びシンクロナイザ２８と接続されている。プロトコル・ユーティリティ３０は、シンクロナイザ・マネージャ２６によって駆動され、ネットワークに対し最も信頼性の高い接続を選択する。

【００８２】動作時には、システム・ユーティリティ又はアプリケーションが、オブジェクトコンテナ２２又はシンクロナイザ・マネージャ２６のいずれかから同期を開始する。シンクロナイザ・マネージャ２６は、プロトコル・ユーティリティ３０に問い合わせて、同期させる２つのサーバ間に高信頼性接続をオープンさせる。その後、シンクロナイザ・マネージャ２６は、プロトコル・ユーティリティ３０からの結果に基づいて、シンクロナイザ２８などのシンクロナイザを生成する。そして、２つのサーバ上のシンクロナイザ２８は、同期化処理を開始する。

【００８３】ここで、図３において、図２のシステムで利用されるサブルーチン又はメソッドについて説明する。オブジェクトコンテナ２２は、putメソッド３２、deleteメソッド３４、getメソッド３６、synchronizeメソッド３８、applyUpdateメソッド４０、generateUpdatesメソッド４２、purgeOffHistoryメソッド４４及びgetSummarizingVersionVectorメソッド４６を利用する。これらのメソッドはJavaによって書かれており、後述するソース・コード中に見つけることができる。

【００８４】図示しているように、オブジェクトコンテナ・マネージャ２４には、openObjectContainerメソッド５０が与えられており、シンクロナイザ２８には、startメソッド５２、pullメソッド５４、pushメソッド５

６及びstopメソッド５８が与えられている。これらはすべてJavaで書かれており、後述するソース・コードに見つけることができる。

【００８５】シンクロナイザ・マネージャ２６には、synchronizerメソッド６０が与えられており、プロトコル・ユーティリティ３０には、標準のisConnectedメソッド６２及びgetBestConnectionメソッド６４が与えられている。

【００８６】ここで、図４において、オブジェクトコンテナ２２に使用される本システムのデータ構造を示す。重要なのは、この発明を通して使用される要約バージョンベクトル（ここでは７０で示す）の構造である。図示しているように、オブジェクトコンテナ２２の要約バージョンベクトルは、識別用フィールド７２及びタイムスタンプ用フィールド７４を有している。Sid_1はシステムのオブジェクトコンテナ１を示し、１５４４という数は、オブジェクトコンテナ１に関連するタイムスタンプを示す。要約バージョンベクトルは、本質的に、個々のコンテナのオブジェクトすべてについての識別子及びタイムスタンプを含んでいる。リモート・オブジェクトコンテナについての同様の要約バージョンベクトルを、７６で示す。このように、システム内の多数のロケーションに、要約バージョンベクトルがあることが分かる。

【００８７】１セットのリモート・オブジェクトコンテナが、共通の差分更新を既に受信しているコンテナを予め定義しているか又は予め選択していることにより、８０で示すように、オブジェクトコンテナから読取ることができる差分更新を識別する役割を果たす、最新共通先祖バージョンベクトルと呼ばれる特殊なバージョンベクトルがある。これによって、いくつかの更新が予め定義されたオブジェクトコンテナのセットに既に格納されているため、その更新をパージすることができる。

【００８８】８２で示すように、オブジェクトコンテナ２２に対する更新の履歴を含むログ構造を示す。図示しているように、ログには、オブジェクト・バージョンベクトル８６又は更新スタンプ８８のいずれかを参照するフィールド８４、及び対応するオブジェクト９２又は差分更新９４を参照するフィールド９０が含まれる。更新９４に関係するオブジェクト９６には、追加の更新９８が含まれており、それらはここで両方ともベース１００で示す初期オブジェクトを更新したものである。

【００８９】オブジェクト９２は、常にオブジェクト全体として送信されなければならないオブジェクトであり、一方オブジェクト９６は、ベースに対する差分更新と言えるオブジェクトである。

【００９０】図５では、同期中にオブジェクトコンテナ内のデータを変更することができるシステムを示している。図示しているように、オブジェクトコンテナ１（ここでは、１１０で示す）には、１１２で示すサーバ１の要約バージョンベクトルVV_1が与えられている。また、オブジェクトコンテナ１１０には、サーバ１以外のサーバの要約バージョンベクトルOVV_1（ここでは１１４で示す）も与えられている。１１６で示すサーバ１の最新共通先祖バージョンベクトルCVV_1もまた、オブジェクトコンテナ１１０に与えられている。同様に、１１８で示すサーバ１の更新ログ１もまた、オブジェクトコンテナ１１０に与えられている。

【００９１】オブジェクトコンテナ１１０’は、指示された入力１１２’～１１８’を有するオブジェクトコンテナ１１０の動的に変化している状態を示している。オブジェクトコンテナ１１０の状態は、オブジェクトコンテナ１２０が、１２０’で示すように動的に変化し、矢印１２０で示すようにオブジェクトコンテナ１１０にデータを送信することによって動的に変化する。同期プロセスは、オブジェクトコンテナ１１０が、矢印１２２で示すように、オブジェクトコンテナ２（ここでは１２０で示す）に関連するサーバに対し要約バージョンベクトル１を送信することによって、開始する。オブジェクトコンテナ１２０は、要約バージョンベクトル１が供給する同期情報を含むように変化し、矢印１２４で示すように、情報を送信することによって、オブジェクトコンテナ１１０を更新する。

【００９２】理解されるように、オブジェクトコンテナ１２０は、入力として、１２６で示すサーバ２の要約バージョンベクトルＶＶ２、１２８で示すサーバ２以外のサーバの要約バージョンベクトル０ＶＶ２、１３０で示すサーバ２の最新バージョンベクトルＣＶＶ２、及び１３２で示すサーバ２の更新ログ２を有する。

【００９３】矢印１２２で示すように、送信される情報は、要約バージョンベクトル２に、オブジェクトコンテナ１２０内の差分同期をサポートするオブジェクトの識別子を足したものである。

【００９４】矢印１３４で示すように、オブジェクトコンテナ１１０’からオブジェクトコンテナ１２０”へ情報が送られることによって、同期は継続する。矢印１３４で示すように転送される情報は、コンテナ１のオブジェクトがない差分同期をサポートするオブジェクトの識別子であるが、そのオブジェクトはオブジェクトコンテナ２内に存在する。これは、矢印１２４で示すようにオブジェクトコンテナ１２０’から受信したオブジェクトの識別子を、オブジェクトコンテナ１内のオブジェクトの識別子と比較することによって、確認することができる。なお、要約バージョンベクトルは、同期プロセスの第１段階で前もって送信されているため、この時点では送信されない。

【００９５】矢印１２４に示すように情報が転送されることにより、オブジェクトコンテナ１２０”は、古い情報をパージすることができる。履歴上古い情報は、最新共通バージョンベクトルＣＶＶ２’（ここでは１３０’で示す）にアクセスすることによって、パージされる。

【００９６】その後、更新は、オブジェクトコンテナ１２０”からオブジェクトコンテナ１１０”に送信され、この時点で、パージ動作を行うことができる。すべてのパージ動作が行われた後、同期は完了し、オブジェクトコンテナ１１０”は、システムのすべてのオブジェクトの最終的に更新されたバージョンを含んでいる状態となる。オブジェクトコンテナ１２０内のオブジェクトもまた、同様に同期させ更新することができることがわかるだろう。

【００９７】図６では、オブジェクトコンテナ内のオブジェクトを変更するシステムを示している。ここで、オブジェクトコンテナ１４０には、要約バージョンベクトル１４２及び更新ログ１４４が与えられている。１４６で示すような変更ステップでは、異なる要約バージョンベクトル（ここでは１４８で示す）と、それに関連する更新ログ（ここでは１５０で示す）を入力する必要がある。この図は、オブジェクトコンテナ１４０が変更された場合、変更を記録するために、対応する要約バージョンベクトルとその対応する更新ログとを変更する必要があることを示している。

【００９８】ここで、図７では、同期プロトコルを示している。同期プロトコルには、２つのシンクロナイザが含まれており、１つは同期開始側であり、もう１つは同期応答側である。同期を開始するシンクロナイザは、まず、同期に応答するシンクロナイザが同期開始側に応答する用意ができた時、そのシンクロナイザに対し、プル／プッシュ（pull/push）要求を送信する。その後、この応答側のサーバは、要約バージョンベクトルを返信する。そして、次の要約バージョンベクトルが、差分同期をサポートするオブジェクトの識別子と共に反対方向に転送される。次に、オブジェクトの識別子が、そのまた反対方向に返送され、その結果、初期同期要求に応答するサーバに対し更新が送信される。

【００９９】同期プロトコルの結果、同期を開始するサーバに最新共通先祖バージョンベクトルに対応する更新が与えられると共に、すべての差分更新がパージされる。同期要求に応答するサーバについても、同様である。

【０１００】図８では、転送すべき更新を抽出する方法を示している。ここでは、サーバ１６０は、上述したフィールド７２、７４を含む関連する要約バージョンベクトル１６２を有している。強調されているフィールドは、第２のサーバ、すなわちサーバ１６４の要約バージョンベクトルにおける対応する値より時間的に早いタイムスタンプを示している。これは、サーバ１６４に関連する要約バージョンベクトル１６８の強調部１６６によって示す。

【０１０１】サーバ１６４の更新ログ１７０におけるバージョンベクトル又は更新スタンプをチェックすることにより、更新が抽出される。なお、これらバージョンベクトル又は更新スタンプは、図４において８６で表すバージョンベクトルである。一実施形態において、チェック・プロセスは以下のように行う。すなわち、要約バージョンベクトル１６２において強調されている対応するタイムスタンプより大きく、かつ、要約バージョンベクトル１６８において強調されている対応するタイムスタンプより小さいか又はそれと等しい、少なくとも１つのタイムスタンプを含むバージョンベクトル又は更新スタンプを有するオブジェクト又は差分更新を、抽出する。

【０１０２】図９では、競合を検出し解消する方法を示し、ここでは、サーバ１８０が対応する要約バージョンベクトルを有している。更新ログ１８４は、サーバ１８０に接続されており、そのベース１９４の最上位に３つの差分更新１８８、１９０、１９２を含むオブジェクト１８６を有している。第２のサーバ、すなわちサーバ１９６には、要約バージョンベクトル１９８及び対応する更新ログ２００が供給される。

【０１０３】サーバ１９６は、１８６との競合を示す異なる差分更新を有するという点を除いてはオブジェクト１８６と同じであるオブジェクト２０２を有する。これは、強調部２０４、２０６を強調部１８８と比較することによって示す。問題は、更新２０４、２０６が更新１８６と競合している理由をどのようにして見つけるかである。このような競合は、オブジェクト全体のバージョンベクトル又は更新スタンプ（この図では図示せず）、あるいはサーバ１８０内の差分更新１８８、１９０、１９２及びサーバ１９６内の差分更新２０４、２０６、２０８、２１０を、要約バージョンベクトル１８２、１９８の共通バージョンベクトル２１２と比較することによって、検出する。２つの要約バージョンベクトルの共通バージョンベクトルは、２つのサーバの共通する状態を反映している。共通バージョンベクトル２１２における対応するタイムスタンプより大きいタイムスタンプを有するサーバ１８０、１９６の双方において、オブジェクト１８６、２０２の差分更新がある場合、２つのオブジェクト１８６、２０２は競合している。

【０１０４】オブジェクト１８６、２０２が競合していることが見つかった後、予め決められた照合調整メソッドを呼出し、２２０で示すメソッドに競合している差分更新を渡すことにより、その競合が解消すなわち照合調整される。なお、照合調整メソッドは、通常、特定のアプリケーションに基づいて決定される。

【０１０５】図１０では、更新ログをパージする１つの方法を示している。ここでは、サーバ２２８は、自身の要約バージョンベクトル（２６０で示す）、及び他のサーバからの要約バージョンベクトル２６２、２６０を有している。サーバ２２８の最新共通バージョンベクトル２３６は、３つのサーバすべての共通の状態を表す３つの要約バージョンベクトル２３０、２３２、２３４から計算される。

【０１０６】最新共通バージョンベクトル２３６を取得した後、更新ログ２３８内の対応する更新ログ・エントリをパージすることが可能となる。これは、括弧２４０でくくられたログ・エントリを削除又はパージすることができることを意味している。これは、これらの更新が、最新共通バージョンベクトル２３６より古い更新スタンプを有しているためであり、これらの更新が、上記３つのサーバすべて既に取込まれているということを反映している。

【０１０７】図１１に示すように、オブジェクト全体の同期と差分同期とを切換えることができる。ここでは、オブジェクト全体の同期を２５０で示し、差分同期を２５２で示す。スイッチ２５４を用いて、どちらの同期プロセスを使用するかを判断する。スイッチ２５４は、以下のように動作するロジック２５６によって制御される。

【０１０８】一実施形態では、差分同期をサポートすることができるオブジェクトを、オブジェクト全体として送信することによって同期させるか、あるいはその差分更新の一部を送信することによって同期させるかを、以下のロジックで判断する。第１に、オブジェクトが第１のサーバにのみ存在し、第２のサーバにない場合、２つのサーバが互いに同期している時、オブジェクト全体を第１のサーバから第２のサーバへ送信する必要がある。

【０１０９】第２に、オブジェクトが２つのサーバに存在する場合であって、少なくとも１つのサーバにおいて、オブジェクトのベースの状態が２つのサーバにおけるオブジェクトの共通する状態より新しい場合、２つのサーバのどちらのオブジェクトも、他方のサーバから差分更新のみを受信した後、最新の状態で同期することができないため、２つのサーバの間でオブジェクトの差分同期を行うことができない。この場合、オブジェクト全体の同期に切換える必要がある。それ以外の場合は、差分更新を２つのサーバの間で実現することができ、システムを差分同期に切換えることができる。

【０１１０】オブジェクト全体の同期と差分同期との切替えのロジックに、必要であれば他の要素を加えることができる。例えば、差分同期からオブジェクト全体の同期への切替えを、メモリの空き領域がほとんど使い果たされている場合、又はその差分更新よりもオブジェクト全体を送信することがより効率的であると思われる場合に、トリガするようにしてもよい。あるいは、オブジェクト全体の同期から差分同期への切替えにより、転送されるデータを最小にすることができる。言換えれば、オブジェクト全体と比べてオブジェクトの差分更新を送信する方が効率的である場合、差分同期に切換える方が望ましい。なお、オブジェクト全体の同期と差分同期との間の切換えは、個々のオブジェクトに基づいて決定される。

【０１１１】特に、本システムで用いられるデータは、

ファイル、文書、データベース又はこれらの一部あるいはすべての混合であってよく、それらは適当に加工される。データをグループ化する方法は、アプリケーション毎に異なる。例えば、ファイル・システムでは、ディレクトリをファイル・グループとしてみなすことができ、データ（ファイル）を編成するために内部の構造をさらに定義することができる。

【０１１２】オブジェクト指向データベースでは、データ・グループに、そのアプリケーションをサポートする必要がある同質又は異質のオブジェクトすべてを含むことができる。第１のステップとして、互いに同期させる必要がある異なる装置のデータ・グループすべてが、結局まったく同じデータ・セットを含むものとする。

【０１１３】本システムは、Javaを使用して実現されている。従って、以下の説明は、Javaの専門用語に基づいている。オブジェクトコンテナは、Javaのオブジェクト・グループを格納及び保持するコンテナである。オブジェクトコンテナ内のオブジェクトは、replace/applyメソッド及びreconcileメソッドを供給するJavaの「直列化可能オブジェクト」である同期可能オブジェクトによって表される。

【０１１４】同期可能オブジェクトは、ファイル、文書又はデータベース・テーブルの行のような具体的なデータを抽象化したものである。（同期可能）オブジェクトの全セットが、あるオブジェクトコンテナ内にすでに存在すると仮定すると、アプリケーションは、そのget又はputメソッドを呼出すことによって、オブジェクトコンテナ内のオブジェクトにアクセスすることができる。

【０１１５】サーバにおいて、オブジェクトコンテナ・マネージャのopenObjectContainerメソッドを呼出すことにより、複数のオブジェクトコンテナを生成することができる。各サーバでは、１つのオブジェクトコンテナ・マネージャしか存在することができず、それを用いてサーバでのオブジェクトコンテナすべてを管理する。オブジェクトコンテナを生成するだけでなく、生成されたオブジェクトコンテナをリスト化するか又は削除するといったサービスもまた提供する。便宜上、各オブジェクトコンテナは、生成された時に、全体的に一意の識別子が割当てられ、その識別子は、本システムのそのオブジェクトコンテナを参照する時に使用することができる。一実施形態では、オブジェクトコンテナの識別子は、入力としてオブジェクトコンテナのＵＲＬを用いて一方向ハッシュ関数によって生成されるハッシュ・コードに割当てることができる。

【０１１６】２つのオブジェクトコンテナを一貫した状態にするために、シンクロナイザは、本システムの重要な要素である。各サーバに、異なる型の通信トランスポート及びプロトコルを使用する複数のシンクロナイザを設けることができる。例えば、シンクロナイザ１は、ＴＣＰ／ＩＰ及び両方向同期プロトコルを使用し、シンク

ロナイザ２は、赤外線プロトコル及び信頼性の低い無線通信により適した一方向プロトコルを使用するようにしてもよい。シンクロナイザは、必要な時にシンクロナイザ・マネージャによって動的に生成され、同期を終了した直後に自己消失する。

【０１１７】サーバ内のシンクロナイザ・マネージャは、シンクロナイザと同期する予定であるオブジェクトコンテナと結合してシンクロナイザが同期を開始するのを止める（trig）ようにしてもよいし、又は、リモート・サーバのシンクロナイザ・マネージャからの１又は複数の同期要求に従う（listen）ようにしてもよい。プロトコル・ユーティリティは、シンクロナイザ・マネージャが、現在使用可能な最良の接続に合致する生成すべきシンクロナイザの適切な選択に役立つ。プロトコル・ユーティリティは、Java拡張パッケージが供給するクラスの１つである。

【０１１８】上述した説明から分かるように、オブジェクトの内容を表すために使用されるフォーマット、及び平行な更新競合を照合調整するために使用されるアルゴリズムのような、オブジェクトのセマンティクスが、シンクロナイザから切離されている。このような特徴により、本システムが、異なるアプリケーション・システム間で同期することができるようになる。

【０１１９】サーバ１が、そのオブジェクトコンテナ１をサーバ２のオブジェクトコンテナ２と同期させるよう指定されているとする。また、オブジェクトコンテナｉ（ｉ＝１、２）にバンドルされた要素は、すべてシンクロナイザ・マネージャｉ、シンクロナイザｉ等のように注釈を付けるものとする。同期プロセス全体の詳細なステップは、以下の通りである。

【０１２０】シンクロナイザ・マネージャ１は、通信チャネルをオープンにし、同期要求に従っているシンクロナイザ・マネージャ２に接続する。シンクロナイザ・マネージャ１は、同期すべきオブジェクトコンテナ及び同期に使用するシンクロナイザを示す同期要求を、シンクロナイザ・マネージャ２に送信する。

【０１２１】シンクロナイザ・マネージャ１、２は、一組の同じシンクロナイザ１、２を作成して接続する。ここで、シンクロナイザ１は同期開始側とし、シンクロナイザ２は応答側とする。

【０１２２】シンクロナイザ１は、シンクロナイザ２に対しpull又はpush更新コマンドを送信する。pull更新コマンドは、オブジェクトコンテナ１がオブジェクトコンテナ２からの更新を受信する必要があることを意味しており、push更新コマンドは、オブジェクトコンテナ１がオブジェクトコンテナ２に対して更新を伝搬することを意味している。シンクロナイザ１がpull又はpush更新コマンドのいずれを送信すべきか、あるいは順次両方を送信すべきかは、同期を開始するシステム・ユーティリティ又はアプリケーションによって判断される。

【０１２３】以下のステップでは、シンクロナイザ１がシンクロナイザ２にpush更新コマンドを送信したと仮定する。シンクロナイザ１がシンクロナイザ２にpull更新コマンドを送信した場合、シンクロナイザ２は、シンクロナイザ１に対して確認メッセージを返送し、残りのステップは上記ステップのまったく逆（鏡像）となる。シンクロナイザ１は、オブジェクトコンテナ１の図３に示すようなgetSummarizingVersionVectorメソッドを呼出すことによって、オブジェクトコンテナ１の要約バージョンベクトルを取得し、シンクロナイザ２に対しその要約バージョンベクトルを送信する。オブジェクトコンテナ１の要約バージョンベクトルは、オブジェクトコンテナ１の現在の状態を反映している。

【０１２４】シンクロナイザ２は、必要な時、受信したオブジェクトコンテナ１の要約バージョンベクトルを、オブジェクトコンテナ２内に記録する。そして、シンクロナイザ２は、図３に示すgetSummarizingVersionVectorメソッドを呼出すと共に、後述するソース・コードに示すオブジェクトコンテナ２のentriesメソッドを呼出すことにより、オブジェクトコンテナ２の要約バージョンベクトルと、オブジェクトコンテナ２内の、差分同期をサポートするオブジェクトの識別子とを取得する。オブジェクトコンテナ２の要約バージョンベクトルは、オブジェクトコンテナ２の現在の状態を反映している。そして、シンクロナイザ２は、上記のように取得した要約バージョンベクトル及び識別子をシンクロナイザ１に送信する。

【０１２５】シンクロナイザ１は、必要な時、受信したオブジェクトコンテナ２の要約バージョンベクトルを、オブジェクトコンテナ１内に記録する。そして、シンクロナイザ１は、受信したオブジェクトコンテナ２の要約バージョンベクトルと受信した識別子とをアーギュメントとして、後述するソース・コードに示すオブジェクトコンテナ１のgetWholeObjectIdsメソッドを呼出すことにより、オブジェクト全体として受信しなければならないオブジェクトの識別子を取得する。

【０１２６】上記のように取得された識別子には、オブジェクトコンテナ２には存在するがオブジェクトコンテナ１には無いオブジェクト、及びベースの状態がオブジェクトコンテナ１及びオブジェクトコンテナ２の共通の状態より新しいオブジェクトの識別子が含まれている。その後、シンクロナイザ１は、上記で取得した識別子をシンクロナイザ２に送信する。

【０１２７】シンクロナイザ２は、受信したオブジェクトの識別子とステップ６で受信したオブジェクトコンテナ１の要約バージョンベクトルとをアーギュメントとして、オブジェクトコンテナ２の図３に示すようなgenerateUpdatesメソッドを呼出すことにより、オブジェクトコンテナ１に送信しなければならない更新を取得する。上記のように取得した更新の各々は、バージョンベクト

ルを有するオブジェクト全体でも、又は更新スタンプを有する差分更新でもよい。そして、シンクロナイザ２は、上記のように取得した更新を１つずつシンクロナイザ１に送信する。シンクロナイザ１は、オブジェクトコンテナ１の図３に示すようなapplyUpdateメソッドを呼出すことにより、受信した更新の各々オブジェクトコンテナ１に与える。

【０１２８】更新を送信又は受信した後、シンクロナイザ１、２は、オブジェクトコンテナ１又はオブジェクトコンテナ２に関係があるすべてのオブジェクトコンテナに含まれている差分更新をパージするために、オブジェクトコンテナ１及びオブジェクトコンテナ２の図３に示すpurgeOffHistoryメソッドを呼出すことができる。

【０１２９】オブジェクトコンテナのapplyUpdateメソッドは、オブジェクトを、受信したオブジェクトに置換えるか、オブジェクトを、受信したオブジェクトと照合調整するか、オブジェクトに受信した差分更新を加えるか、又は受信した差分更新をそれと競合するオブジェクトの差分更新と照合調整するか、いずれかのために、オブジェクト内で定義されたメソッドの１つをさらに呼出す。また、それは、オブジェクトコンテナが既にその中に更新を配置したという事実を表すために、オブジェクトコンテナの要約バージョンベクトルを更新する。

【０１３０】上述した説明から分かるように、本システムで展開する要約バージョンベクトルは、並行な更新競合の検出にのみ使用するのではなく、更新ログの中のエントリの数を減らすためにも使用する。

【０１３１】オブジェクトコンテナの要約バージョンベクトルは、実質的に、オブジェクトコンテナ内のすべてのオブジェクトのバージョンベクトル及びすべての差分更新の更新スタンプを要約したものである。差分同期をサポートすることができないオブジェクトのバージョンベクトルは、オブジェクト全体の現在の状態を反映し、差分同期をサポートすることができるオブジェクトのバージョンベクトルは、関連する差分更新がすべて無いオブジェクトのベースの現在の状態を反映する。オブジェクト全体についてのバージョンベクトルのように、オブジェクトのベースのバージョンベクトルもまた、関連する差分更新の一部又は全部がパージされるのに従って動的に変化する。従って、オブジェクトのバージョンベクトルは、オブジェクト全体又はオブジェクトのベースの現在の状態を特徴付ける。

【０１３２】オブジェクトコンテナの要約バージョンベクトル又はオブジェクトのバージョンベクトルは、更新スタンプのベクトルである。更新スタンプは、２つのフィールドからなる。すなわち、第１のフィールドは、ベクトルがオブジェクトのバージョンベクトルである場合はオブジェクトに対し、又はベクトルがオブジェクトコンテナの要約バージョンベクトルである場合はオブジェクトコンテナ内のオブジェクトに対し、変更を行ったオブジェクトコンテナの識別子である。

【０１３３】また、第２のフィールドは、オブジェクトに対し最後の変更を行った時、第１のフィールドに対応する変更するオブジェクトコンテナを変更することによって生成されたタイムスタンプである。便宜上、更新スタンプは（sid_i、t_i）で表す。ここで、sid_iは更新スタンプの第１のフィールドであり、t_iは更新スタンプの第２のフィールドである。更新スタンプ内のタイムスタンプは、実時間、更新一連番号、又はアプリケーションにとって意義深いものにすることができる。

【０１３４】オブジェクトコンテナが生成する一連のタイムスタンプは、意味が一貫しており、まったく単調でなければならないが、異なるオブジェクトコンテナは、それらが生成するタイムスタンプを有する異なるセマンティクスを有することができる。これは、異なるアプリケーション・システムが同じセマンティクスでタイムスタンプを生成することを必要とするのは実際的でないため、異なるアプリケーション・システムに亙ってオブジェクトコンテナ間の同期をとる場合に、非常に重要である。

【０１３５】上述した更新スタンプの表現を用いて、オブジェクトコンテナの要約バージョンベクトル又はオブジェクトのバージョンベクトルを、{(sid_1、t_1)、　、(sid_i、t_i)、　、(sid_n、t_n)}又は{(sid_i、t_i)|i=1、2、　、n}と表すことができる。オブジェクトコンテナの要約バージョンベクトル及びオブジェクトのバージョンベクトルのセマンティクスは、オブジェクトのバージョンベクトルが、それを包含しているオブジェクトコンテナの現在の状態ではなく、包含しているオブジェクトコンテナ内のオブジェクトの現在の状態のみを反映しているという点を除けば、同様である。これらが同様であるため、要約バージョンベクトルについてのみ以下に詳細を説明する。オブジェクトのバージョンベクトルは、必要な時にのみ言及する。

【０１３６】オブジェクトコンテナの要約バージョンベクトルのｉ番目の要素は、オブジェクトコンテナが、タイムスタンプt_iが示す時間まで識別子sid_iを有するオブジェクトコンテナが行う変更をすべて含んでいることを意味する。便宜上、オブジェクトコンテナｊの要約バージョンベクトルをvv^jと表し、そのｉ番目の更新スタンプを$vv_i{}^j$と表す。

【０１３７】オブジェクトコンテナｊの要約バージョンベクトルvv^jのサイズｎは、ｊ番目のオブジェクトコンテナが、その中のオブジェクトに対して、タイムスタンプt_1、t_2、　、t_nまで識別子sid_1、sid_2、　、sid_nを有するオブジェクトコンテナが行う変更をそれぞれすべて含んでいることを意味する。上述した説明に基づき、要約バージョンベクトル、バージョンベクトル及び更新スタンプを利用して、２つのエンティティの状態を比較することができるように、以下にいくつかの関数／命題を

定義／証明する。以下、エンティティは、オブジェクトコンテナ、オブジェクト又は差分更新を表す。

【０１３８】定義：z=y.getTime(x)は、識別子ｘを有するオブジェクトコンテナに対応するｙのタイムスタンプを戻す関数である。ｙは、オブジェクトコンテナの要約バージョンベクトル、オブジェクトのバージョンベクトル又は差分更新の更新スタンプを表している。ｘがｙで表されない場合、戻される更新スタンプｚは０である。

【０１３９】定義：y_1及びy_2を、オブジェクトコンテナの要約バージョンベクトル、オブジェクトのバージョンベクトル又は差分更新の更新スタンプのいずれかとする。y_1及びy_2内のオブジェクトコンテナの識別子についてy_1.getTime(x)=y_2.getTime(x)である場合、y_1は、y_2と等しい。y_1及びy_2内のオブジェクトコンテナの識別子についてy_1がy_2と等しくなく、かつy_1.getTime(x)>=y_2.getTime(x)である場合、y_1は、y_2より新しい。y_1及びy_2内のオブジェクトコンテナの識別子についてy_1がy_2と等しくなく、かつy_1.getTime(x)<=y_2.getTime(x)である場合、y_1は、y_2より古い。y_1がy_2と等しくなく、y_2より新しくなく、かつy_2より古くない場合、y_1はy_2と競合する。

【０１４０】命題：e_1及びe_2は、オブジェクトコンテナ、オブジェクト、オブジェクトのベース又は差分更新のいずれかを表すものとし、y_1及びy_2は、e_1及びe_2それぞれの要約バージョンベクトル、バージョンベクトル又は更新スタンプであるものとする。y_1がy_2と等しい場合、e_1の現在の状態は、e_2と同じか又は一致する。y_1がy_2より新しい場合、e_1の現在の状態はe_2より新しい。y_1がy_2より古い場合、e_1の現在の状態は、e_2より古い。y_1がy_2と競合する場合、e_1の現在の状態は、e_2と競合する。

【０１４１】証明：定義により、y_1がy_2と等しい場合、e_1に含まれる更新はすべてe_2にも含まれ、その逆でもある。明らかに、y_1がy_2と等しい場合、e_1の現在の状態はe_2と同じである。次に、y_1がy_2より新しく、e_1の現在の状態がe_2より新しくないものとする。定義により、y_1がy_2より新しい場合、e_1は、e_2に含まれていない更新を含んでいなければならず、e_2に含まれる更新はすべてe_1に伝搬されていなければならない。そのため、上記仮定は妥当ではない。命題の他の部分も同様に証明することができる。（結果を証明。）

【０１４２】定義：y_1及びy_2は、それぞれオブジェクトコンテナe_1及びe_2の要約バージョンベクトルであるものとする。r_yが、y_1及びy_2の両方に存在するオブジェクトコンテナの識別子のみを含み、そのような識別子の各々に対応するr_y内のタイムスタンプが、y_1及びy_2内の対応するタイムスタンプの小さい方である場合、y_1及びy_2の共通のバージョンベクトルをr_yと呼ぶ。

【０１４３】命題：e_1及びe_2は、２つの異なるオブジェクトコンテナであるものとし、y_1及びy_2は、それぞれy_1

及びy_2の要約バージョンベクトルであるものとする。さらに、r_yは、y_1及びy_2の共通のバージョンベクトルであるものとする。従って、r_yは、e_1及びe_2がそこから分かれた状態を表す。

【０１４４】証明：y_1及びy_2の定義により、更新がe_1及びe_2の両方に含まれている場合、それがy_1及びy_2の両方に反映されていなければならず、そのため、更新に関係するタイムスタンプは、y_1及びy_2内の対応するタイムスタンプと等しいかそれより小さくなければならない。言換えれば、更新に関係するタイムスタンプは、y_1及びy_2内の２つの対応するタイムスタンプの小さい方と等しいかそれより小さい。共通のバージョンベクトルの定義により、更新はr_yに反映される。

【０１４５】更新がe_1又はe_2のいずれかにのみ含まれる場合、更新に関係するタイムスタンプは、他方のオブジェクトコンテナの要約バージョンベクトルにおける対応するタイムスタンプより大きくなければならない。従って、更新は、更新の無いオブジェクトコンテナの要約バージョンベクトルによって反映され得ない。また、共通のバージョンベクトルの定義により、更新は、r_yによっても影響され得ない。（結果を証明。）

【０１４６】オブジェクトコンテナに対するオブジェクトの変更を行う度に、オブジェクトコンテナの図３に示すputメソッドを呼出すことにより、その変更はオブジェクトコンテナに反映されることが知られている。そのputメソッドにより、オブジェクトコンテナは、その要約バージョンベクトルを更新するか、変更すべきオブジェクトのバージョンベクトルを更新するか、又は加えるべき差分更新に関係する更新スタンプを生成し、変更したオブジェクト又は加えた差分更新に対する参照、及び対応するバージョンベクトル又は更新スタンプに対する参照を、オブジェクトコンテナの更新ログに加える。差分更新の更新スタンプは、差分更新が、更新スタンプ内の第２のタイムスタンプに示す時刻に、更新スタンプの第１のフィールドに示す識別子で、オブジェクトコンテナによって生成されたことを意味する。

【０１４７】上述した説明に基づき、あるオブジェクトコンテナの他のオブジェクトコンテナとの差を理解し、２つのオブジェクトコンテナ間の並行な更新競合を検出し、更新ログ内のエントリの数を減少させることが可能である。I_sが、R_sで示す他のオブジェクトコンテナｊから更新を取出す（pull）オブジェクトコンテナｉを示すものとする。上述した計算による具体的なアルゴリズムは、以下の通りである。

【０１４８】差分計算　要約バージョンベクトルsvv^jを有するR_sが、I_sの要約バージョンベクトルsvv^iと、オブジェクト全体としてI_sに送信する必要のあるオブジェクトの識別子E^iとを受信したものとする。そして、svv^i及びsvv^jの共通のバージョンベクトルrvv^{ij}は、上記定義に基づいて計算することができ、I_sに送信する必要のあ

る更新はすべて以下のように決定される。

【０１４９】第１に、差分同期をサポートしないオブジェクトについて、オブジェクトのバージョンベクトルをチェックする。そのバージョンベクトルがrvv^{ij}より新しい場合、オブジェクト全体がI_sに対して更新として送信される。U1を、上記計算によって得られる更新のグループとする。

【０１５０】第２に、差分同期をサポートするオブジェクトについて、オブジェクトのベースのバージョンベクトルをチェックする。そのベースのバージョンベクトルがrvv^{ij}より新しい場合、オブジェクト全体がI_sに対して更新として送信される。U2を、上記計算によって得られる更新のグループとする。

【０１５１】次に、差分同期をサポートするオブジェクトについて、それに差分更新が与えられている場合、その差分更新の各々の更新スタンプをチェックする。オブジェクトのそのような差分更新の更新スタンプのうち、rvv^{ij}より新しいか競合しているものがあり、そのオブジェクトの識別子がE_iに含まれている場合、そのオブジェクト全体がI_sに更新として送信される。あるいは、差分更新の更新スタンプがrvv^{ij}より新しいか競合している場合、オブジェクトの差分更新がI_sに対して更新として送信される。U3を、上記計算によって得られる更新のグループとする。

【０１５２】最後に、Uを、U１、U2及びU3の集合体とする。Uにおけるすべての更新の順序は、ある更新が、そのバージョンベクトル又は更新スタンプが他の更新のバージョンベクトル又は更新スタンプより新しい場合、その第１の更新は第２の更新より前にくる、といった順序である。Uの中の２つの更新が競合するバージョンベクトル又は更新スタンプを有する場合、２つの更新が差分更新であるならば、２つの更新の間の順序は、R_sの更新ログにおける順序と一貫していなければならず、そうでなければ、v１及びv2をそれぞれ２つの更新のバージョンベクトル又は更新スタンプとした場合、２つの更新は以下のように順序付けられる。

【０１５３】続いて、最初から最後までｖ１内のオブジェクトコンテナの識別子sliを取出して、条件の１つを満たすsliが見つかるまで以下に示す条件を調べる。v１.getTime(sli)>v2.getTime(sli)のようなsliが存在する場合、v2を有する更新の前にｖ１を有する更新を配置する（put）。ｖ１.getTime(sli)<v2.getTime(sli)のようなsliが存在する場合、v2を有する更新の後でｖ１を有する更新を配置する。同じオブジェクトコンテナが生成したタイムスタンプの特徴が単調であるため、同じオブジェクトコンテナ内に同じバージョンベクトルは２つないことから、最終的に条件の１つが満たされる。

【０１５４】そして、Uは、I_sに送信される、I_sのR_sとの違いを表す更新のセットである。並行競合の検出　I_s及びR_sにそれぞれ含まれる２つのオブジェクト間の並行な更新競合は、I_sが２つのオブジェクトに関する更新を受信した後に検出される。

【０１５５】受信した更新が、そのバージョンベクトルと共にオブジェクト全体である場合、受信した更新のバージョンベクトルは、I_s内のオブジェクトのバージョンベクトルと比較される。受信したオブジェクトがI_s内のオブジェクトのバージョンベクトルより新しいか又は古いバージョンベクトルを有する場合、２つのオブジェクト間に競合はなく、受信したオブジェクトは、I_s内のオブジェクトを置換えるか、又は捨てられる。２つのオブジェクトのバージョンベクトルが競合する場合、２つのオブジェクトは競合しており、I_s及びR_s内のオブジェクトの予め定義されたreconcileメソッドが呼出される。

【０１５６】受信した更新がその更新スタンプを有する差分更新である場合、I_s内の対応するオブジェクトに関する差分更新は、I及びRの要約バージョンベクトルの共通するバージョンベクトルrvv^{ij}と比較される。I_s内のオブジェクトの差分更新に、rvv^{ij}より新しいか又はそれと競合しているものが無い場合、I_s内のオブジェクトとR_s内の対応するオブジェクトの間に競合は無い。そうでなければ、rvv^{ij}より新しいか又はそれと競合するI_s内のオブジェクトの差分更新はすべて、受信した更新と競合し、その競合を解消するためにI_s内のオブジェクトのreconcileメソッドに渡される。

【０１５７】更新ログ削減　第１のオブジェクトコンテナを呼出したIs及びRsの各々について、第２のオブジェクトコンテナを呼出した他方のオブジェクトコンテナが、第１のオブジェクトコンテナの更新ログ削減のプロセスにおいて考慮される第１のオブジェクトコンテナが選択したオブジェクトコンテナのセットの中にある場合、要約オブジェクトコンテナの要約バージョンベクトルは、第１のオブジェクトコンテナ内に記録され、更新ログ削減の以下のステップが実行される。

【０１５８】まず、第１のオブジェクトコンテナに記録された他のオブジェクトコンテナの要約バージョンベクトルがチェックされる。第１のオブジェクトコンテナが選択したすべてのオブジェクトコンテナの要約バージョンベクトルが記録されると、第１のオブジェクトコンテナの最新共通バージョンベクトルが計算されるか又は再計算される。ここで計算された第１のオブジェクトコンテナの最新共通バージョンベクトルには、第１のオブジェクトコンテナに記録された他のオブジェクトコンテナの要約バージョンベクトル及び第１のオブジェクトコンテナ自身の要約バージョンベクトルの各々に含まれるオブジェクトコンテナ識別子のみが含まれる。

【０１５９】第１のサーバの最新共通バージョンベクトルにおける各タイムスタンプは、第１のオブジェクトコンテナに記録された他のオブジェクトコンテナの要約バージョンベクトル、及び第１のオブジェクトコンテナ自身の要約バージョンベクトルにおける対応するタイムス

タンプの最も小さいものである。次に、第１のオブジェクトコンテナの最新共通バージョンベクトルが更新される場合、第１のオブジェクトコンテナの更新ログ内のすべてのログ・エントリがスキャンされ、更新、すなわち、オブジェクト全体、及びバージョンベクトル又は更新スタンプが第１のオブジェクトコンテナの最新共通バージョンベクトルより古いか又は等しい差分更新の記録の削除部分が、第１のオブジェクトコンテナの更新ログからパージされる。

【０１６０】上述した定義から分かるように、第１のオブジェクトコンテナの最新共通バージョンベクトルは、第１のオブジェクトコンテナが選択した他のオブジェクトコンテナすべてと第１のオブジェクトコンテナ自身とに共通するオブジェクトコンテナの最新状態を表す。従って、第１のオブジェクトコンテナの更新ログで更新をパージするセマンティクスは、第１のオブジェクトコンテナが選択した他のオブジェクトコンテナすべてに伝搬された更新が、第１のオブジェクトコンテナから結果的に削除される、ということである。これにより、概して、第１のオブジェクトコンテナが使用するメモリ及びディスク領域の一部を解放することができる。

【０１６１】ここでは、この発明のいくつかの実施の形態について説明してきたが、当業者にとって、上述した内容が、単なる例として提示されているものであって、単に例示的であり限定しているのではないことは、明らかである。多数の変更及び他の実施の形態が、当業者の範囲内にあり、この発明と均等な特許請求の範囲によって定義される発明の範囲内にあるよう企図される。

【０１６２】

【発明の効果】以上のように、この発明によれば、広域モバイルコンピューティングに適応し、サーバを同期させ、同時にプロセスをより効率的にするための汎用システムを提供することができる。

【０１６３】また、二次サーバが概して信頼性の低いサーバによりリンクされた同期プロセスのバックボーンを形成する、一次サーバの高性能高信頼性リンクを用いたネットワークを備えることで、モバイルコンピュータからの同期を、クライアント／サーバ・モード及びピア・ツー・ピアの双方で行うことができ、二次サーバのいかなるトポロジをもサポートすることができる。

【０１６４】また、一次サーバは自動的に、かつ頻繁に同期をとるが、二次サーバの同期は、意図しない同期を避けるユーザの制御下で行われ、要約バージョンベクトルを使用して、オブジェクト毎にバージョンベクトルを交換する必要を無くすことにより、転送されるデータの量を最小にすることができる。

【０１６５】また、この要約バージョンベクトルにより、要約バージョンベクトル及び更新スタンプを使用する差分同期、サーバ上の差分更新をパージするための最新共通バージョンベクトルの生成、影響を受けていない

サーバからのデータを用いた前の失敗地点からの同期の再開、及び転送されるデータのアトムとして差分更新を可能にすることによるきめの細かい同期が可能になる。

【０１６６】さらに、オブジェクト全体の同期と差分同期とを自動的に切換える。また、本システムでは、更新が標準フォーマットで抽出され標準プロトコルに基づいて同期するため、データのセマンティクスが同期化から切離されていることから、異なるシステム間の同期が可能となる。

【図面の簡単な説明】

【図１】　一次サーバ／二次サーバ構造を説明するこの発明のシステム・アーキテクチャを表す図である。

【図２】　オブジェクトコンテナ・マネージャ、同期マネージャ、シンクロナイザ及びオブジェクトコンテナを示す、図１のシステムのサーバのアーキテクチャを示す図である。

【図３】　図２に示す各構成要素についての主なサブルーチンを示す図である。

【図４】　図２のオブジェクトコンテナの構造を示すブロック図であり、その中で用いられる要約バージョンベクトル、最新共通先祖バージョンベクトル及び本システム内で用いられる差分ベースのログの構造を示す図である。

【図５】　要約バージョンベクトルの利用を説明する２つのサーバの同期を示す図である。

【図６】　オブジェクトコンテナ内でオブジェクトを変更するプロセスを示すブロック図である。

【図７】　第１及び第２のサーバ間でのデータの交換を示す同期プロトコルを示す図である。

【図８】　第２のサーバについての要約バージョンベクトル及び差分更新ログを利用して、第１のサーバから第２のサーバへ転送される更新を抽出するシステムを示すブロック図である。

【図９】　２つのサーバ間の競合を検出し解消するプロセスを示すブロック図である。

【図１０】　サーバにおいて更新ログをパージするプロセスを示すブロック図である。

【図１１】　システムの予め決められた特徴に基づき、オブジェクト全体の同期プロセスから差分同期プロセスへ同期システムを切換えることを説明するブロック図である。

【符号の説明】

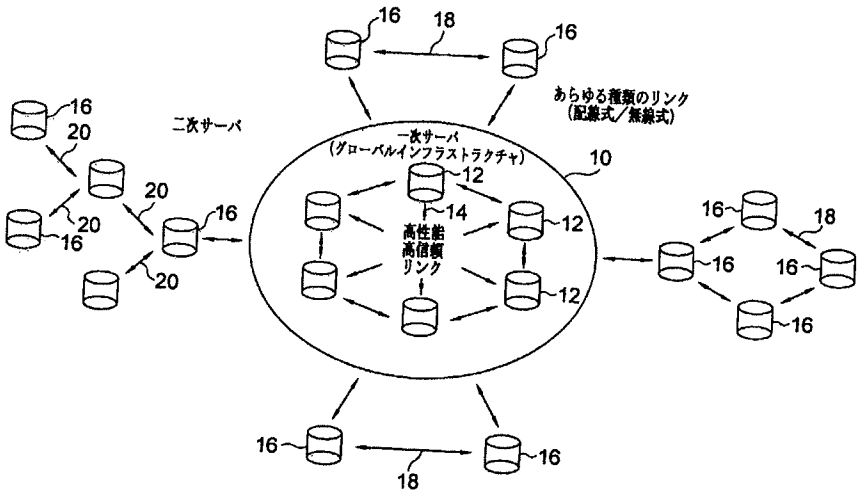
１０　ネットワーク、１２　一次サーバ、１４　高性能リンク、１６　二次サーバ、２２　オブジェクトコンテナ、２４　オブジェクトコンテナ・マネージャ、２６　シンクロナイザ・マネージャ、２８　シンクロナイザ、３０　プロトコル・ユーティリティ、７０　要約バージョンベクトル、７２　識別用フィールド、７４　タイムスタンプ用フィールド、８０　最新共通先祖バージョンベクトル　８２　ログ　９２　オブジェクト　９４　差

39

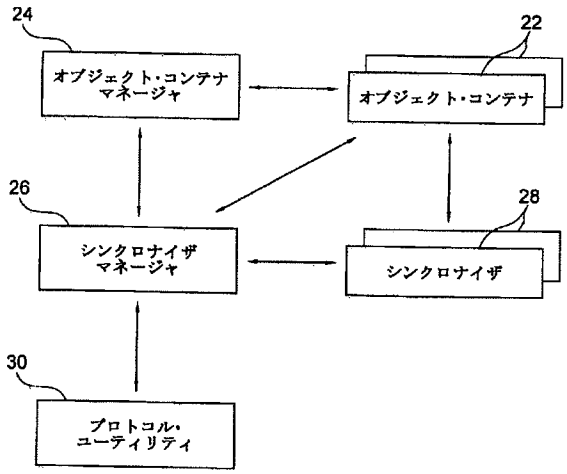
分更新、１１０　オブジェクトコンテナ１、１２０　オブジェクトコンテナ２、１４０　オブジェクトコンテナ、１６０　サーバ１、１６４　サーバ２、２５０　オブジェクト全体の同期、２５２　差分同期、２５４　スイッチ、２５６　ロジック。

40

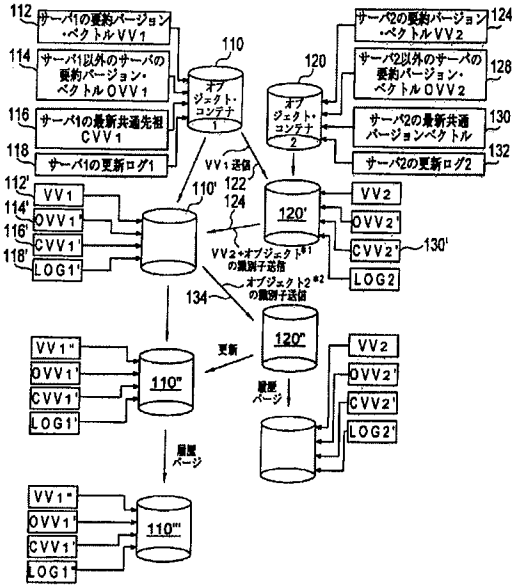
【図１】



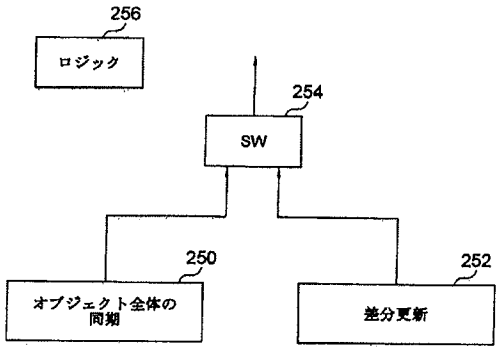
【図２】



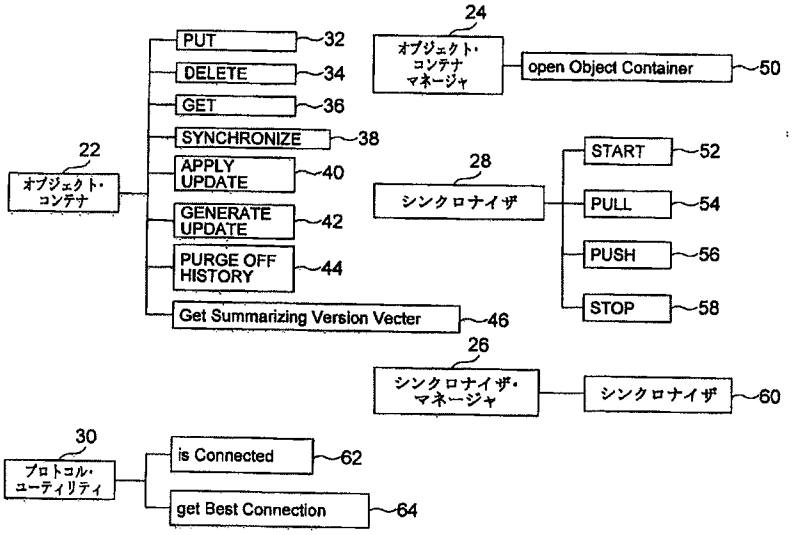
【図５】



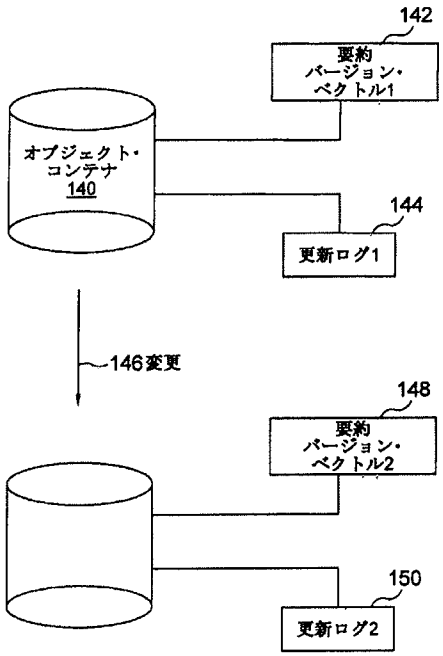
【図１１】



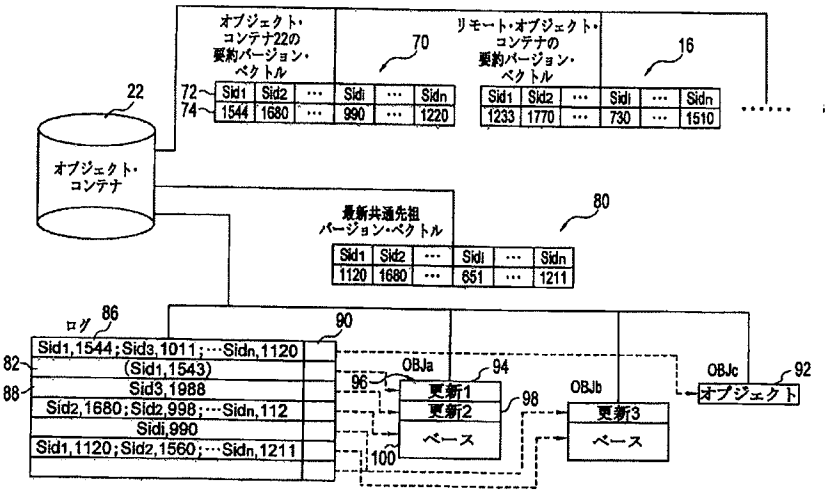
【図３】


22
オブジェクト・コンテナ
PUT
32
DELETE
34
GET
36
SYNCHRONIZE
38
APPLY UPDATE
40
GENERATE UPDATE
42
PURGE OFF HISTORY
44
Get Summarizing Version Vecter
46
24
オブジェクト・コンテナマネージャ
open Object Container
50
28
シンクロナイザ
START
52
PULL
54
PUSH
56
STOP
58
26
シンクロナイザ・マネージャ
シンクロナイザ
60
30
プロトコル・ユーティリティ
is Connected
62
get Best Connection
64

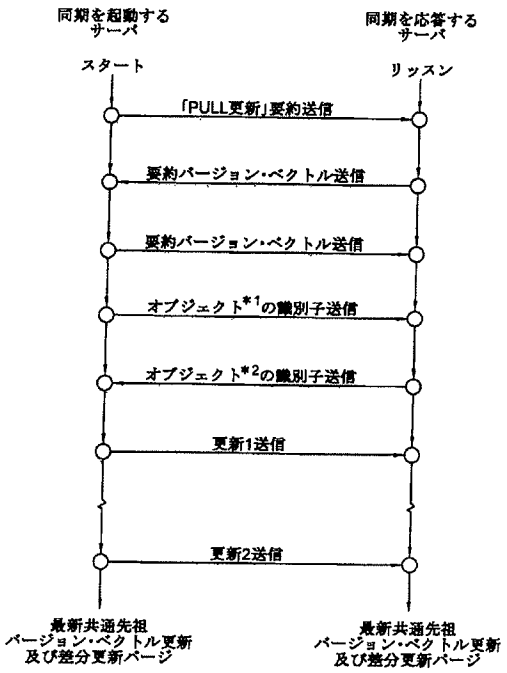
【図６】


142
要約バージョン・ベクトル1
オブジェクト・コンテナ
140
144
更新ログ1
146変更
148
要約バージョン・ベクトル2
150
更新ログ2

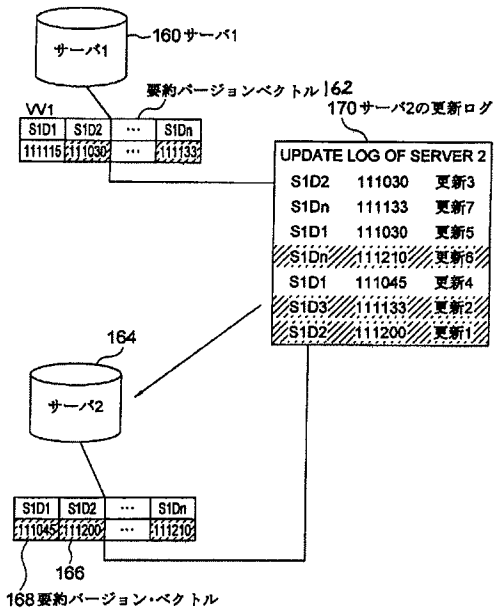
【図４】


オブジェクト・コンテナ22の要約バージョン・ベクトル
70
リモート・オブジェクト・コンテナの要約バージョン・ベクトル
16
22
72
74
Sid1 Sid2 … Sidi … Sidn
1544 1680 … 990 … 1220
1233 1770 … 730 … 1510
オブジェクト・コンテナ
最新共通先祖バージョン・ベクトル
80
1120 1680 … 651 … 1211
ログ
86
90
82
88
Sid1,1544;Sid3,1011;…Sidn,1120
(Sid1,1543)
Sid3,1988
Sid2,1680;Sid2,998;…Sidn,112
Sidi,990
Sid1,1120;Sid2,1560;…Sidn,1211
OBJa
94
96
更新1
更新2
ベース
98
OBJb
更新3
ベース
OBJc
92
オブジェクト
100

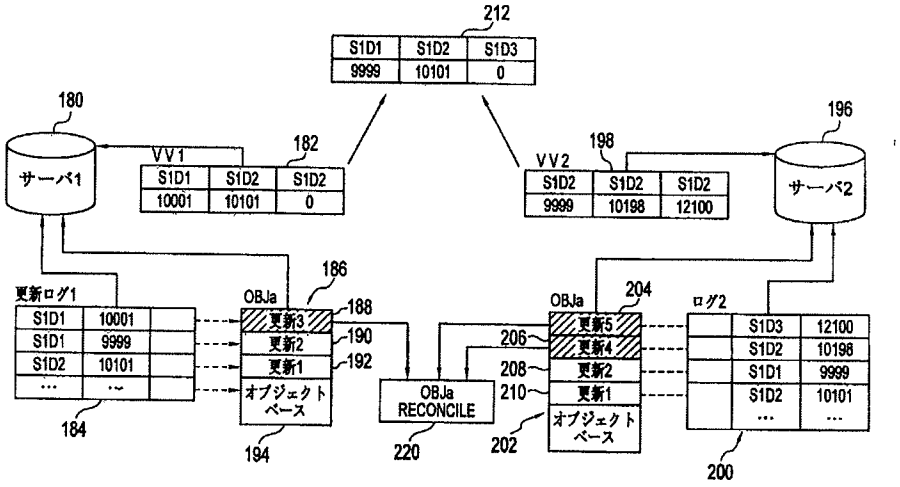
【図７】


同期を起動する
サーバ
同期を応答する
サーバ
スタート
リッスン
「PULL更新」要約送信
要約バージョン・ベクトル送信
要約バージョン・ベクトル送信
オブジェクト*1の識別子送信
オブジェクト*2の識別子送信
更新1送信
更新2送信
最新共通先祖
バージョン・ベクトル更新
及び差分更新ページ
最新共通先祖
バージョン・ベクトル更新
及び差分更新ページ

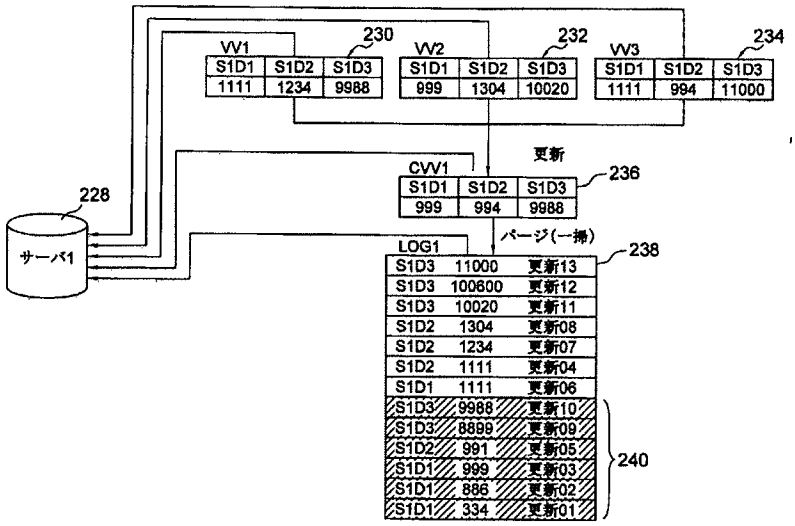
【図８】


サーバ1
160サーバ1
要約バージョンベクトル162
170サーバ2の更新ログ
VV1
S1D1 S1D2 … S1Dn
111115 111030 … 111133
UPDATE LOG OF SERVER 2
S1D2 111030 更新3
S1Dn 111133 更新7
S1D1 111030 更新5
S1Dn 111210 更新6
S1D1 111045 更新4
S1D3 111133 更新2
S1D2 111200 更新1
164
サーバ2
S1D1 S1D2 … S1Dn
111045 111200 … 111210
166
168要約バージョン・ベクトル

【図９】


212
S1D1 S1D2 S1D3
9999 10101 0
180
サーバ1
VV1
182
S1D1 S1D2 S1D2
10001 10101 0
VV2
198
S1D2 S1D2 S1D2
9999 10198 12100
196
サーバ2
更新ログ1
S1D1 10001
S1D1 9999
S1D2 10101
184
OBJa
186
更新3
更新2
更新1
オブジェクト
ベース
188
190
192
194
OBJa
RECONCILE
220
OBJa
204
更新5
更新4
更新2
更新1
オブジェクト
ベース
206
208
210
202
ログ2
S1D3 12100
S1D2 10198
S1D1 9999
S1D2 10101
200

【図１０】



フロントページの続き

(71)出願人 597067574
201 BROADWAY, CAMBRIDGE, MASSACHUSETTS 02139, U. S. A.

(72)発明者 ルウシェン・ペン
アメリカ合衆国、カリフォルニア州、サン・ホセ、サン・トーマス・アキノ・ロード 129、アパートメント・ナンバー 211

(54)【発明の名称】 複数のサーバでオブジェクトを同期させる汎用システム、２つのサーバでオブジェクトを同期させるシステム、２つのサーバでオブジェクトを同期させる方法、２つのサーバに存在するオブジェクトがオブジェクトのタイプに関係なく同期される汎用同期システム、２つのサーバでのオブジェクトの同期における競合を検出して解消する方法